Ibis Data Serialization in Apache Spark

By Dadepo Aderemi and Mathijs Visser
Supervisors:
dr. Jason Maassen (eScience Center)
Adam Belloum (UvA)

We live in a big data world

- Increase in data generation: IoT, mobile devices, social media, logs from large scale software etc.
- Large and complex data sets
- Beyond ability of traditional software tools.
- Rich analytical potential

Image source: https://towardsdatascience.com/what-is-big-data-lets-answer-this-question-933b94709caf

We live in a big data world

- Big data is essential not only in business but in Science
- Computational Astrophysics, Climate Modeling, Medical and Pharmaceutical research etc.
- Volume 455 Issue 7209, 4 September 2008 of Nature magazine talked about the challenges of dealing with big data.
- Core problem: Explosion of data that cannot be managed speedily using traditional approaches.

Big data is high-volume, high-velocity and/or high-variety information assets that demand cost-effective, innovative forms of information processing that enable enhanced insight, decision making, and process automation.

- *Gartner Glossary*

Big data is high-volume, high-velocity and/or high-variety information assets that ***demand cost-effective, innovative forms of information processing*** that enable enhanced insight, decision making, and process automation.

- *Gartner Glossary*


APACHE
Spark
TM

What is Apache Spark

- Is a unified analytics engine for large-scale data processing written in Scala
- Began at UC Berkeley in 2009, Apache project in 2013
- Supports the MapReduce programming model
- Supports both batch and streaming processing of data
- Provides SQL, Machine learning and Graph processing capabilities
- Provides a distributed computing platform that can be run Apache Mesos, Kubernetes, standalone, or in the cloud.
- Has ability to access data in:
  - HDFS (Hadoop Distributed File System)
  - Alluxio, Apache Cassandra, Apache HBase, Apache Hive, and hundreds of other data sources

Common bottleneck in big data processing

- Network bandwidth
- Disk IO
- Memory
- **Serialization**

> *“...the mechanism for converting (graphs of) data (Java objects) to some format that can be stored or transferred (e.g., a stream of bytes, or XML)...”*

Research Questions

- Can Apache Spark's performance be improved by taking advantage of Ibis' serialization techniques?

Sub questions:

- What components of Apache Spark can benefit from Ibis' fast serialization?
- How can Ibis' serialization techniques be integrated into Apache Spark?
- How does the performance of Apache Spark differ when using Java, Kryo and Ibis serialization?


ibis

What is Ibis

- Ibis is an open source Java distributed computing software project
- Developed at the Vrije Universiteit Amsterdam
- With the goal of creating an efficient Java-based platform for distributed computing.[1]

[1] https://www.cs.vu.nl/ibis/

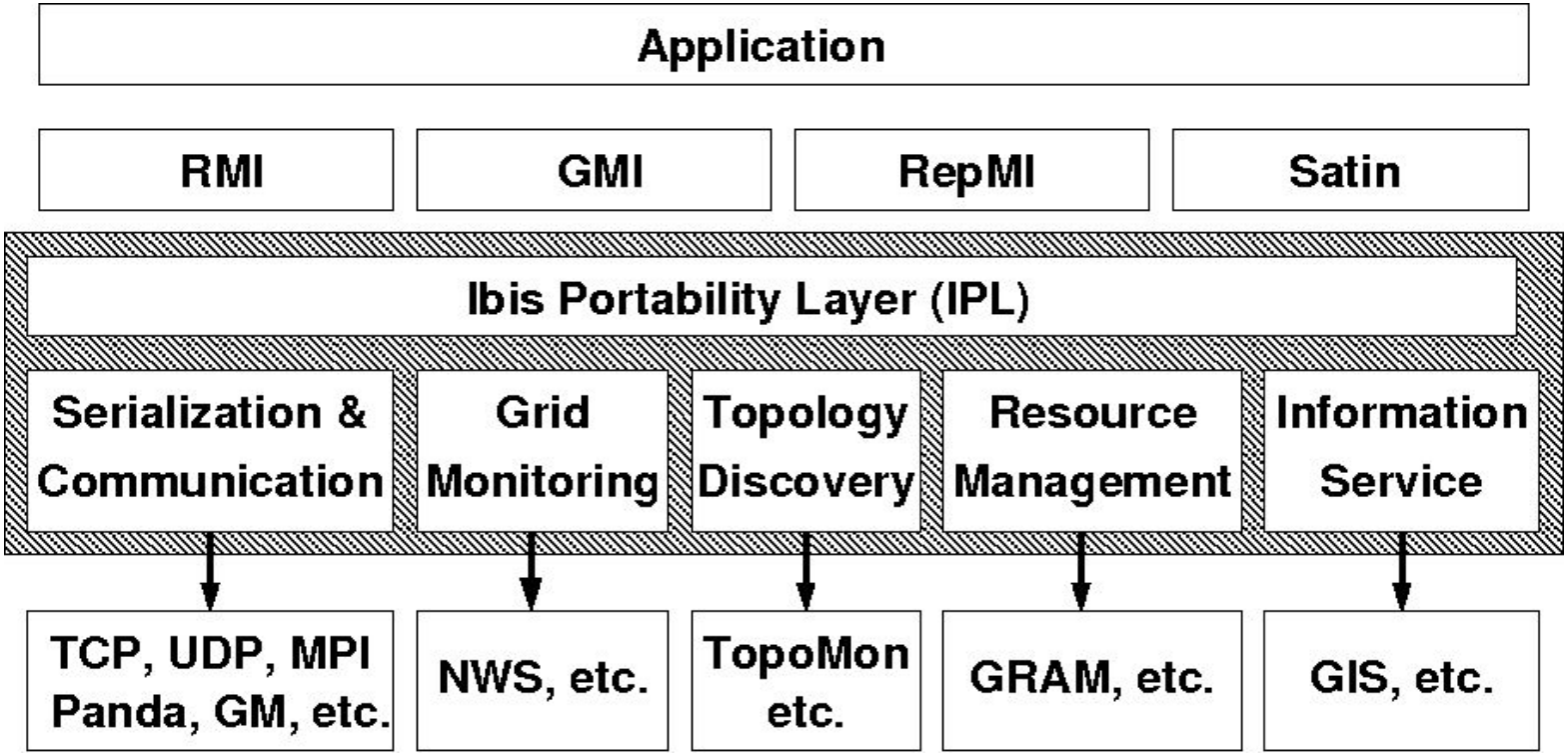
Related work

- Xiaoyi Lu et al.
  - Improvements to Spark has been made using various methods such as Remote Direct Memory Access (RDMA)
  - Applying zero-copy buffer management in the network stack
- van Nieuwpoort, Rob et al
  - Applied compile-time code generation to improve Java's RMI in Ibis RMI
- Apache Spark has also shown serialization performance can be improved using Kryo serialization.

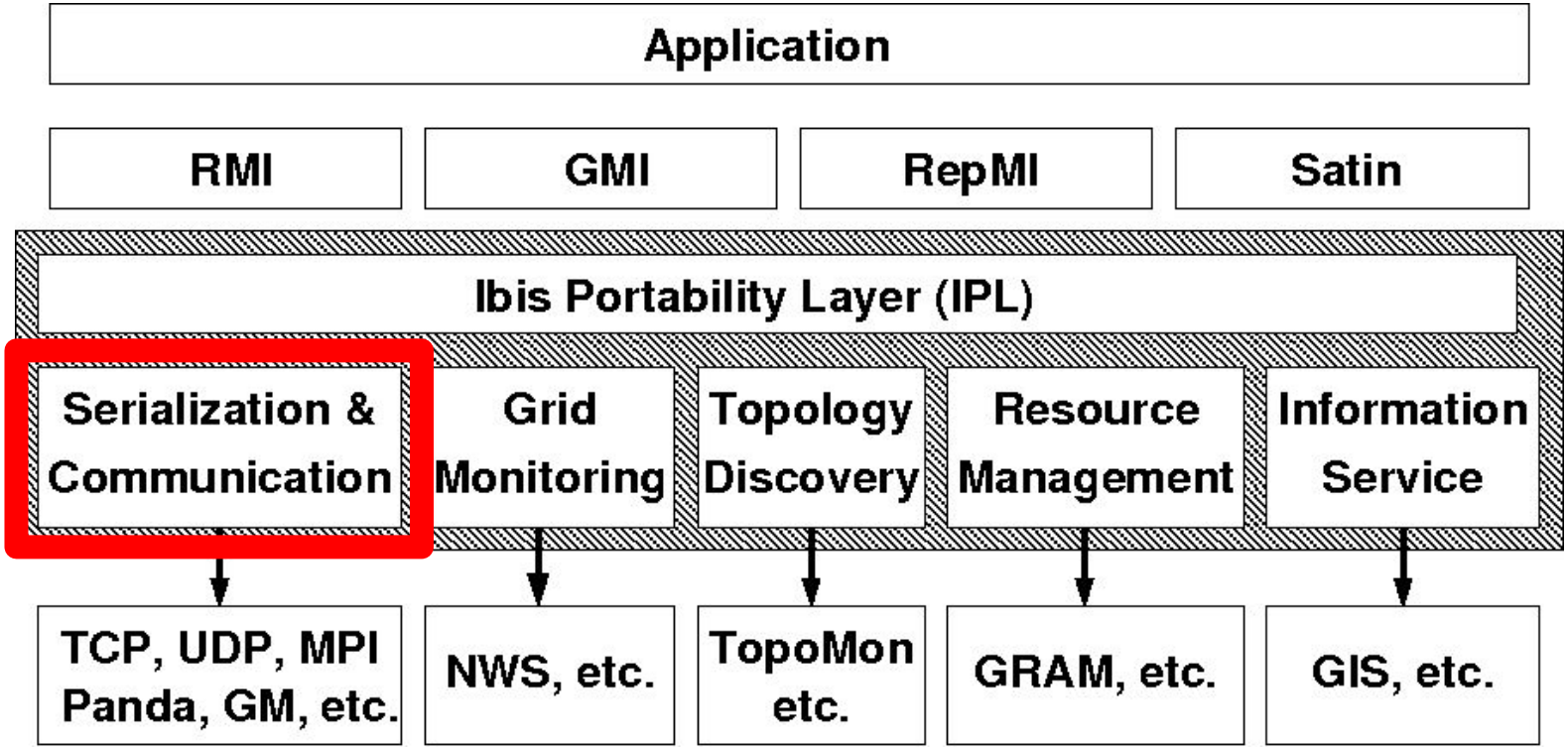
- But no prior work has been done regarding using Ibis serialization in Spark

Overview of Ibis components

What is Ibis software stack: Component view



What is Ibis software stack

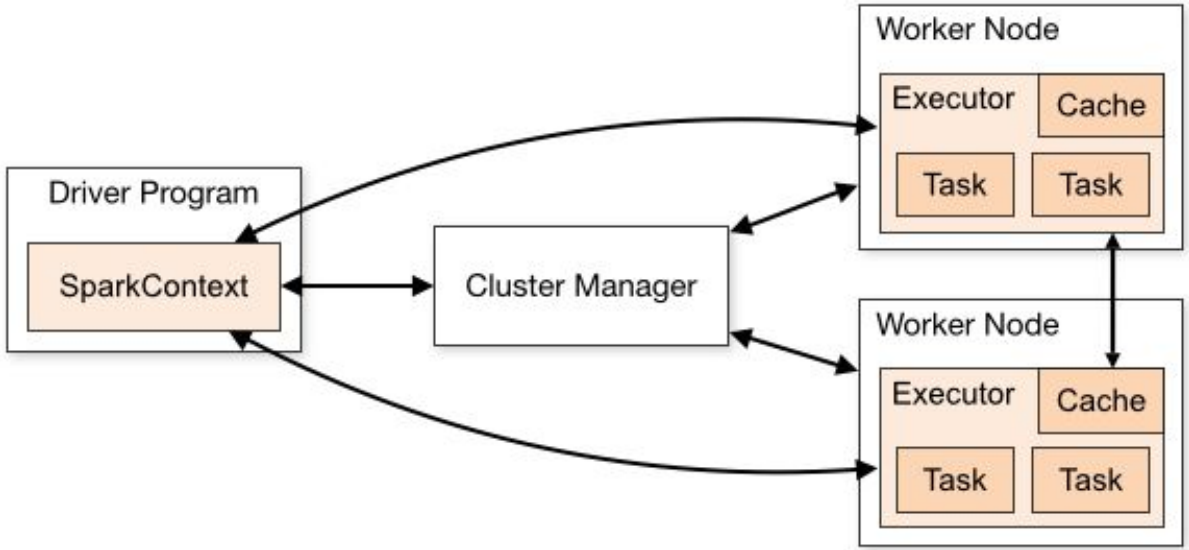


What makes Ibis serialization efficient

- Ibis serialization optimizes:
  - Optimizes object creation
  - Avoiding Data Copying
  - Optionally moves runtime type inspection to compile time

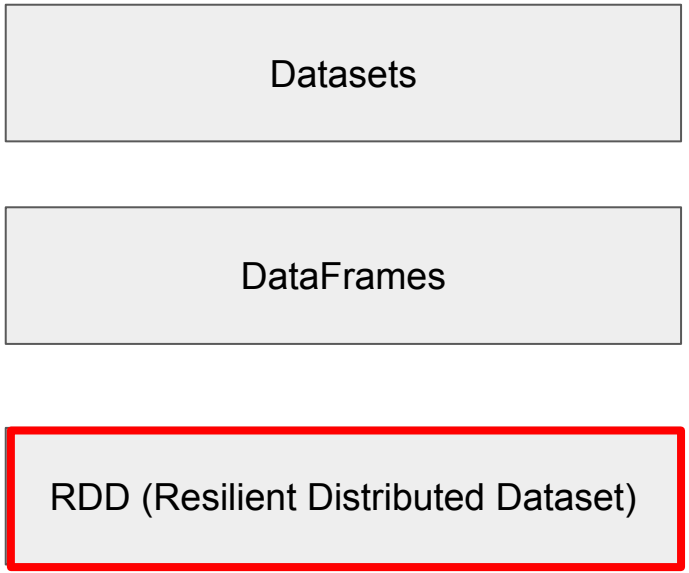
Overview of how Spark works

How Spark Works

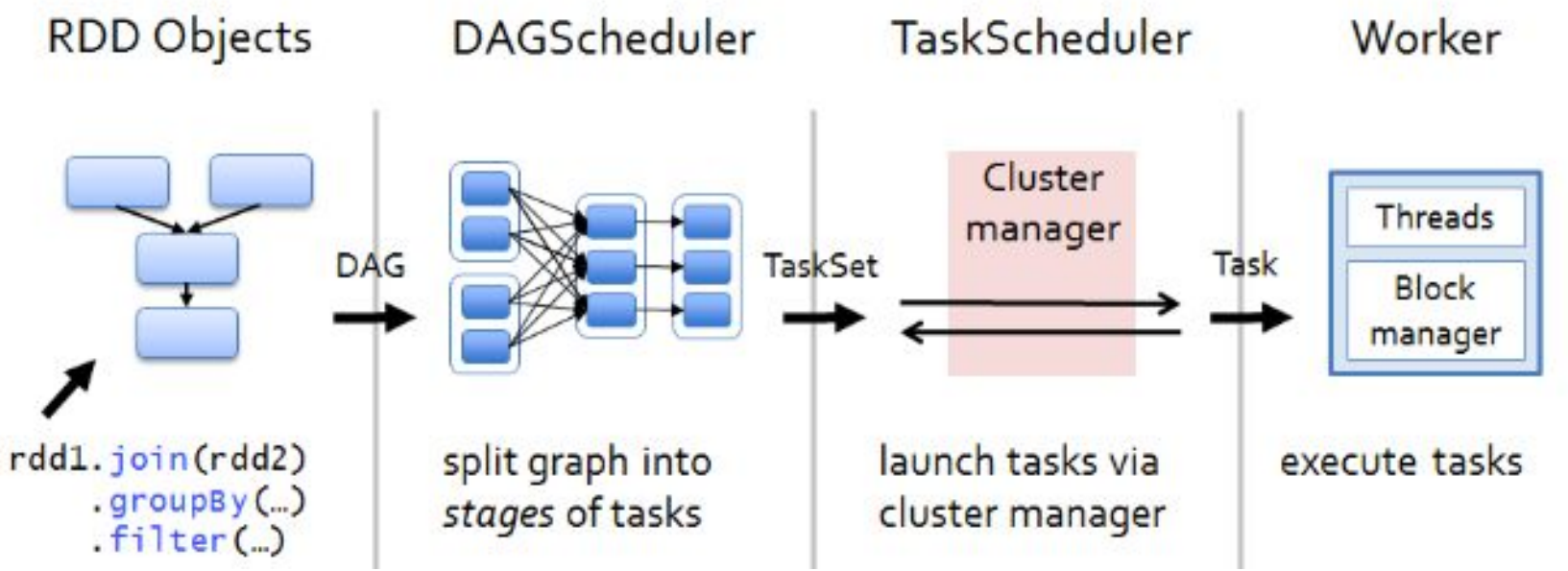


Source: https://spark.apache.org/docs/latest/cluster-overview.html

Spark APIs



How Spark executes applications



Source: https://trongkhoanguyen.com/spark/understand-rdd-operations-transformations-and-actions/

Methodology

Methodology

- Identifying Spark components using serialization.
- Extracting the serialization component in Ibis
- Modify spark to use the serialization from Ibis
- Measure performance difference

Identifying Spark components using serialization

- We analysed the source code of Spark
- We found 17 instances of direct serialization calls
  - Internal operations
  - Network operations
  - Persistence operations (Disk and Memory)
- Available serialization mechanisms:
  - Native Java serialization
  - Kryo serialization [1]

[1] https://github.com/EsotericSoftware/kryo

Modifying Spark to use Ibis serialization

- 17 different components using serialization.
- We managed to replace 15 of those.

Unresolved Incompatibilities.

- Incompatibility with NettyBlockRpcServer and NettyBlockTransferService
  - Uses Zero-copy I/O
  - Off heap network buffer management
  - Making a drop in replacement harder
- Incompatibility with deserializing from Hadoop filesystem.

Resolved Incompatibilities.

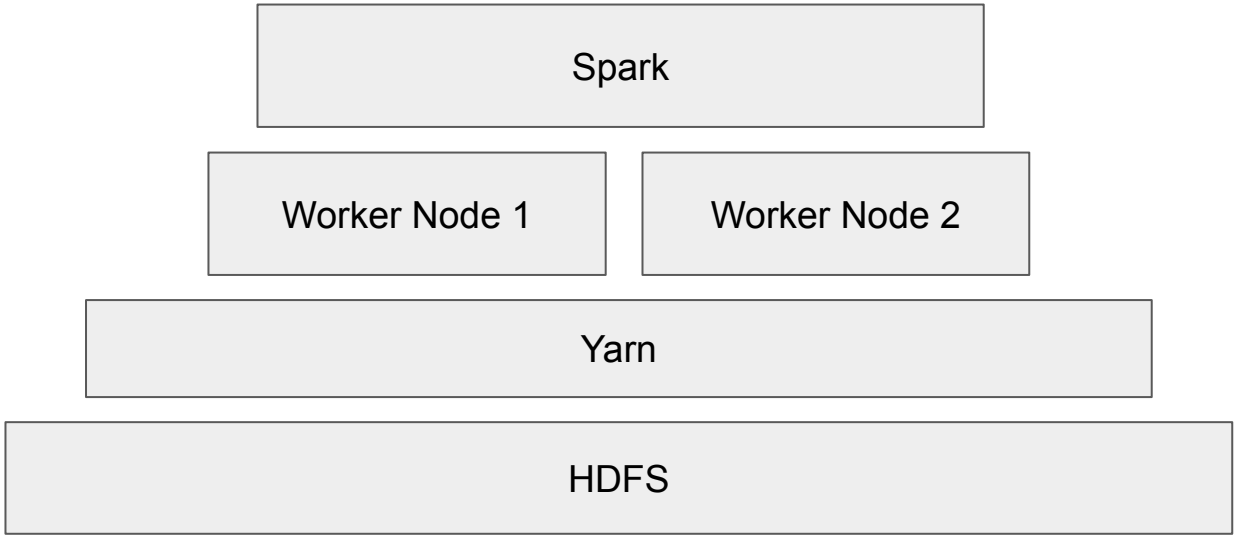
- Modification to support serialization of Scala’s Option type
- Modification to support serialization of Enum with constant method
  - *Thanks to the Ibis maintainer: Ceriel Jacobs from the Vrije University Amsterdam*
- Modification to support ByteBuffer

Measuring the performance differences

Benchmark setup

- We now have a:
  - A modified version of Spark
  - Original Spark version to test Kryo and Native Java serialization
- Two worker nodes, directly connected
- Both running a HDFS DataNode
- Using Hadoop Yarn as resource manager

Benchmark setup



Benchmarking method

- Single test results may not be conclusive
- To get more reliable results we perform each benchmark 50 times
- Take the mean of all results
- Test environments are reset between test runs
- Also comparing Ibis and Ibisc

Benchmark types

- Mostly use standardized benchmarks

- TeraSort:
  - Distributed sorting algorithm
  - Measures shuffling performance
- SparkPi:
  - Computes an approximation of Pi
  - Measures computing performance
- Memory persistence
  - Measure memory persistence performance

Results

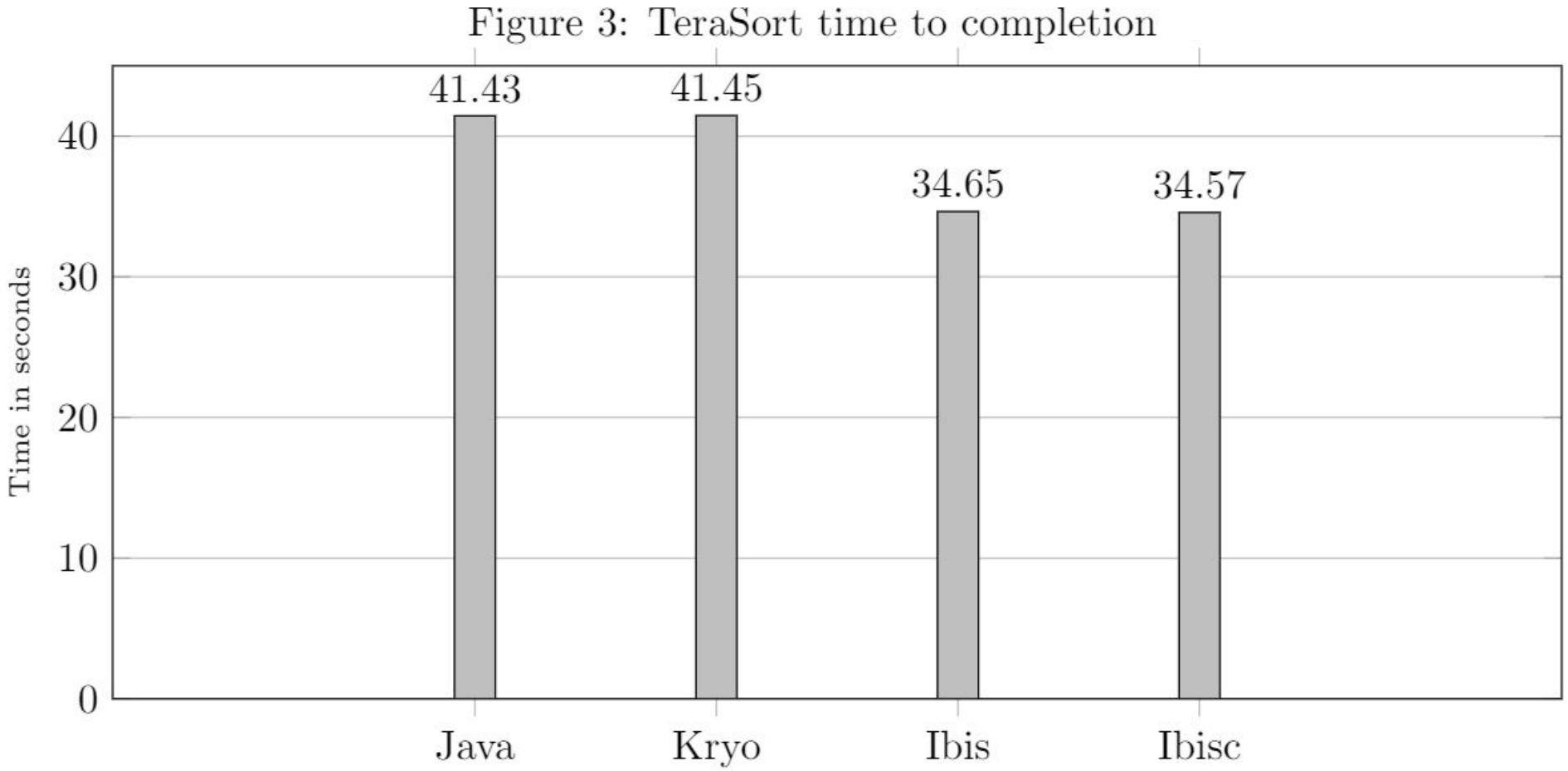
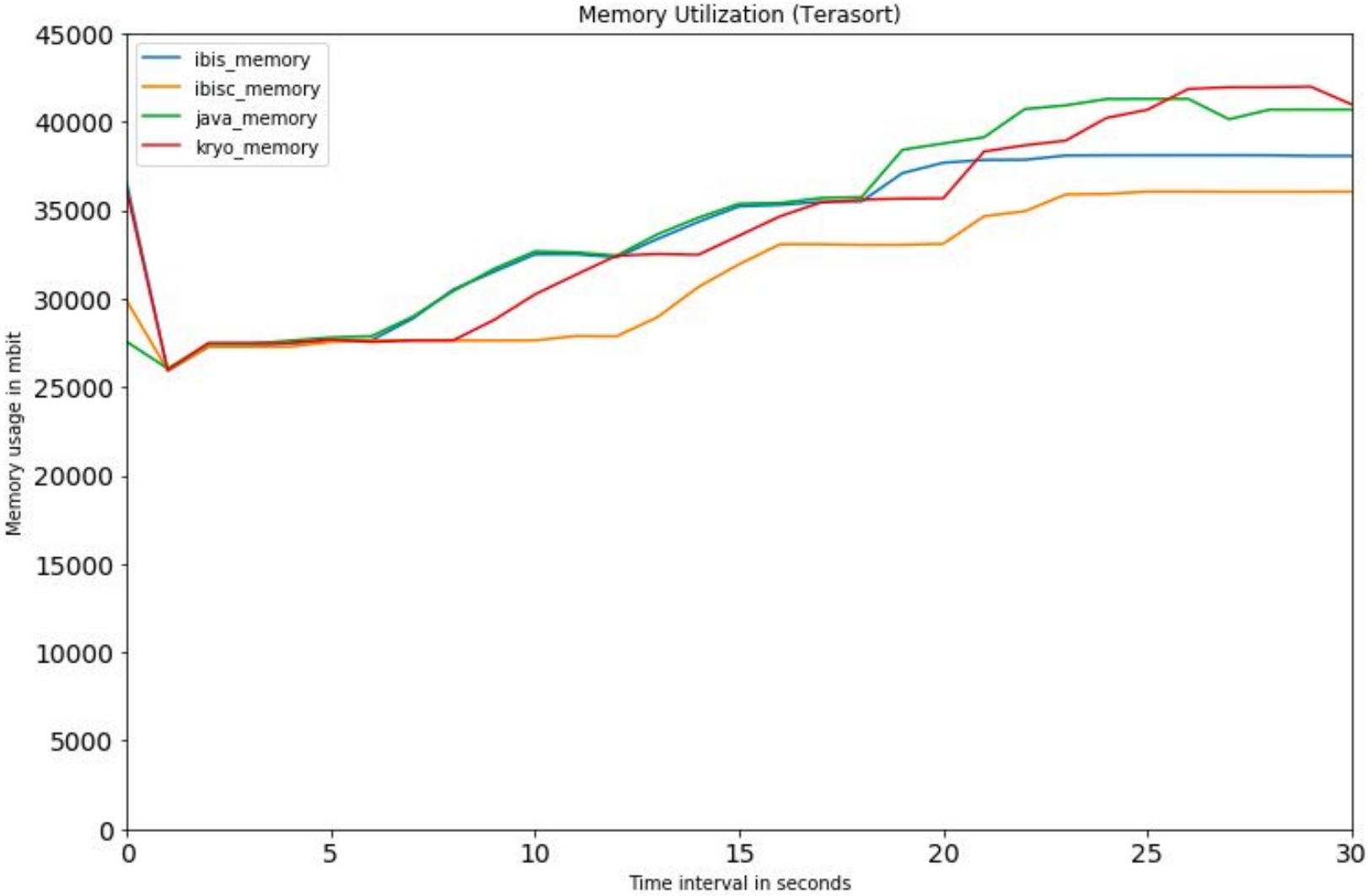
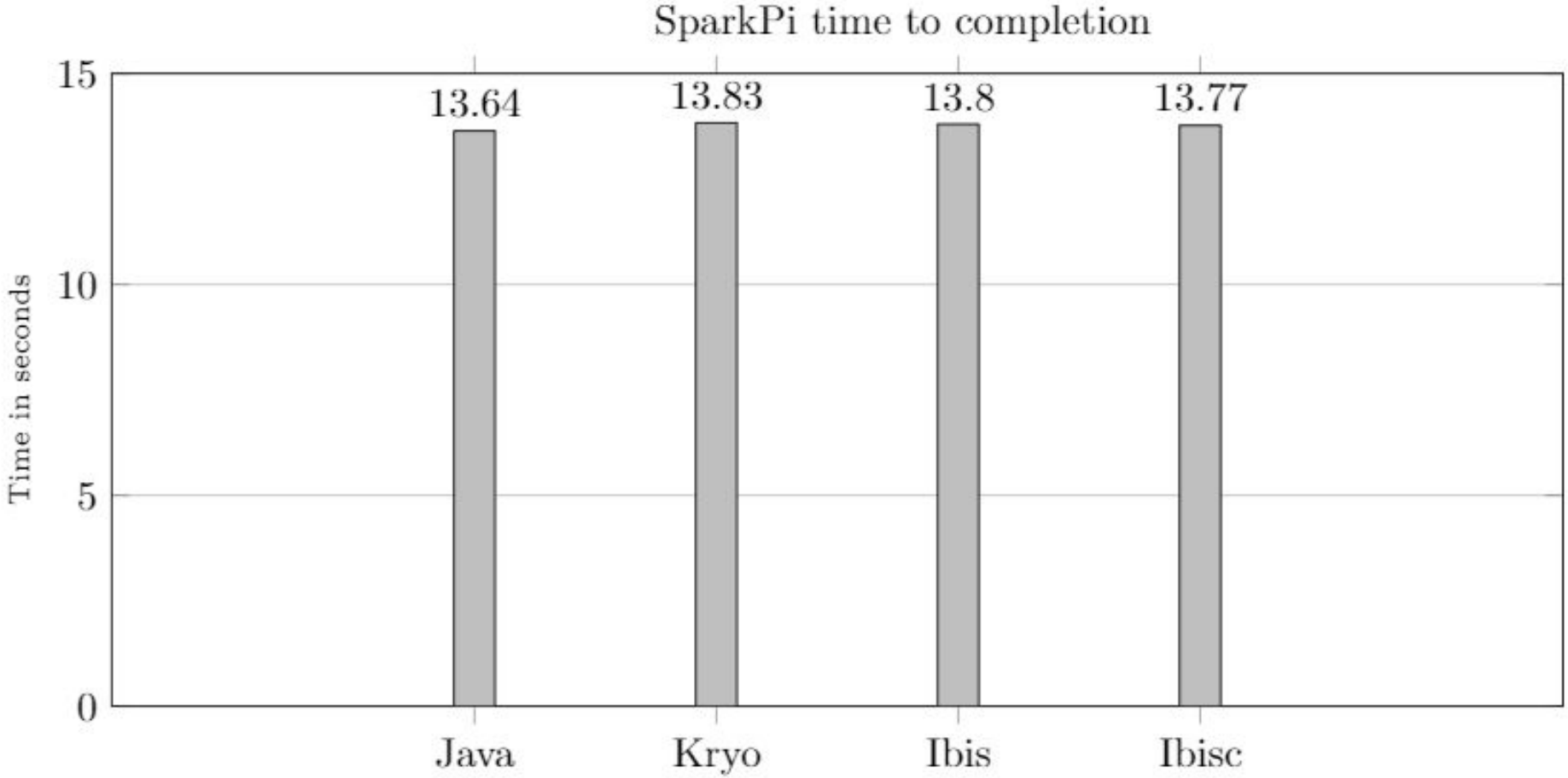
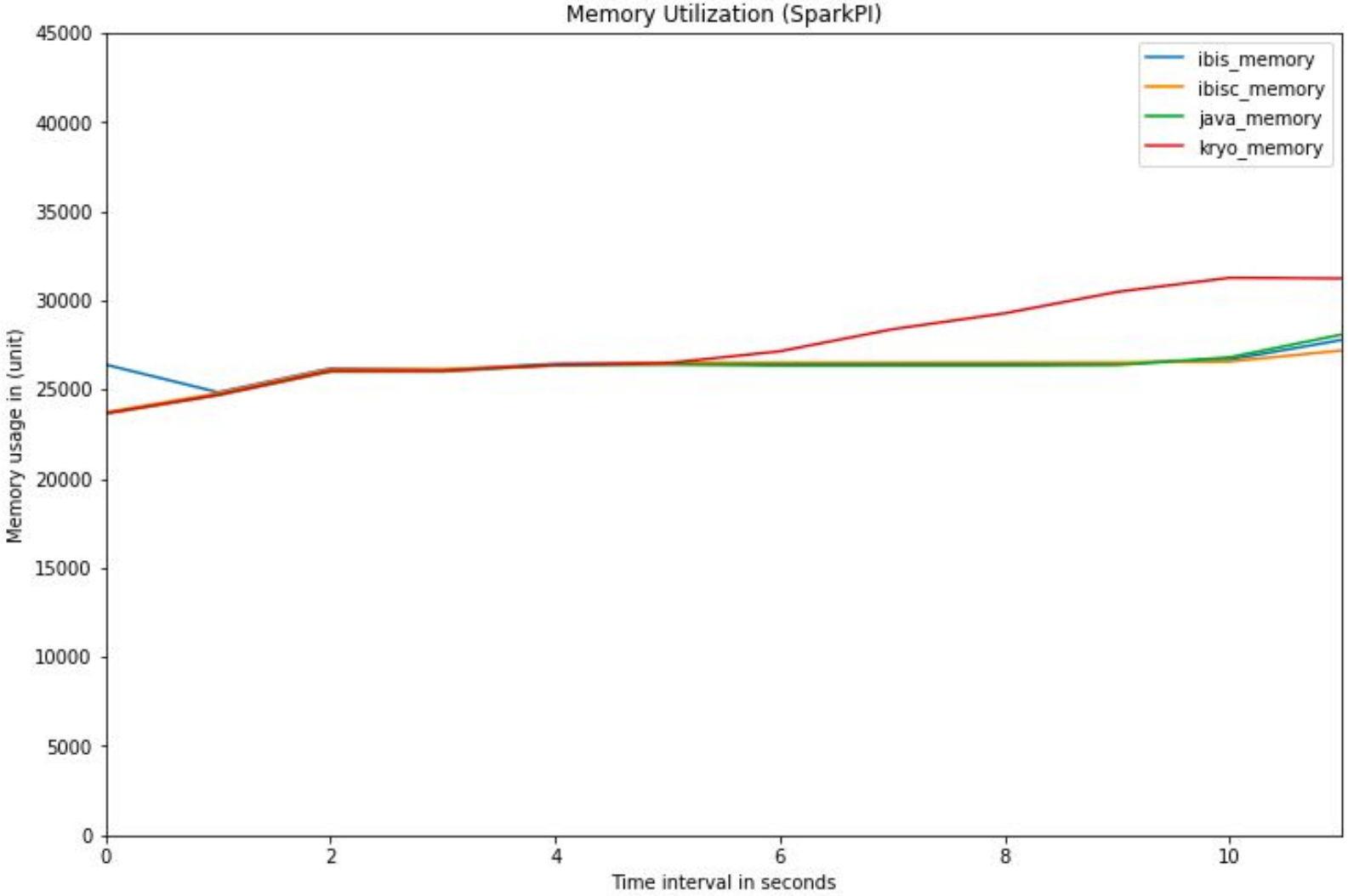
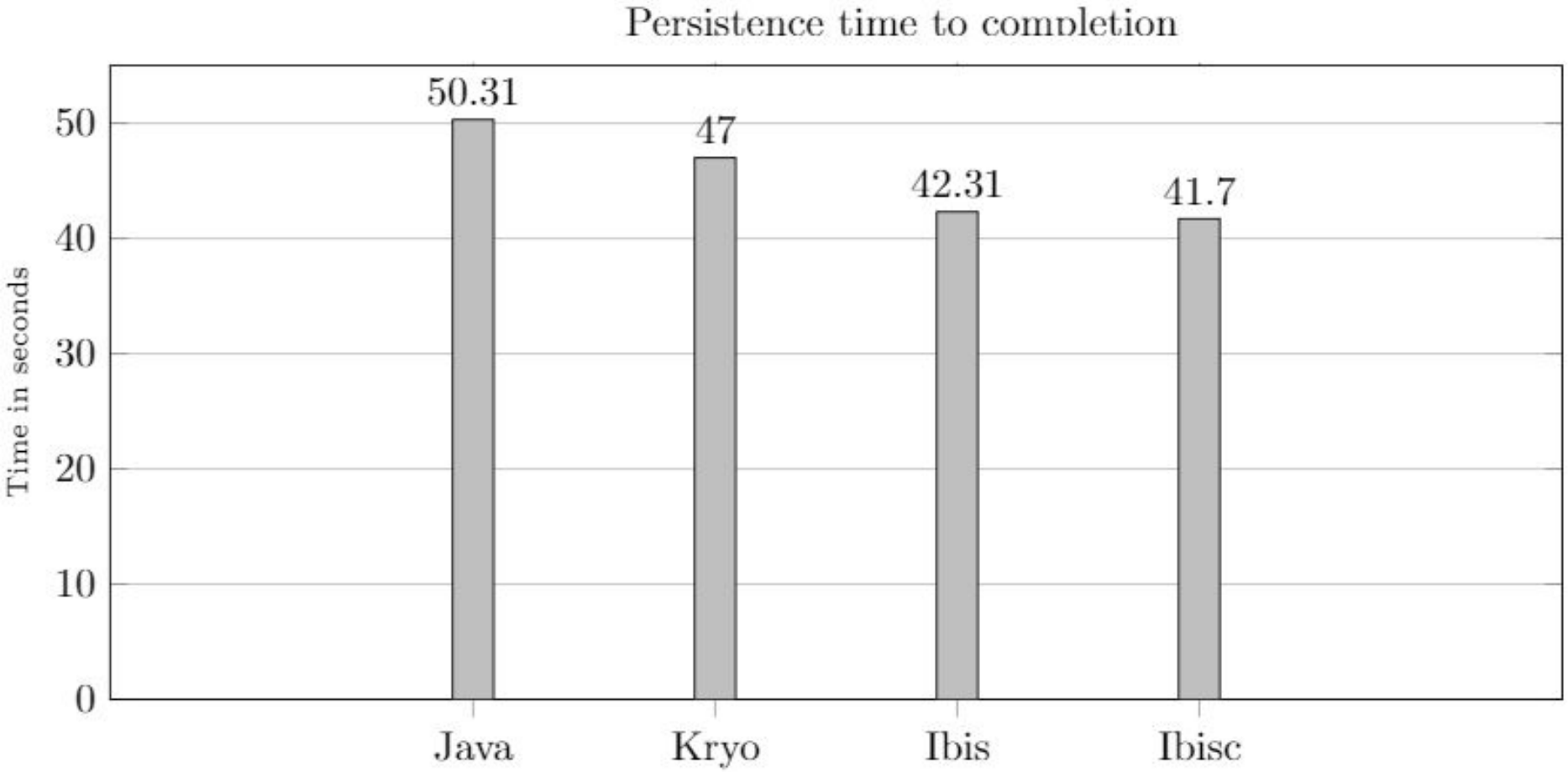


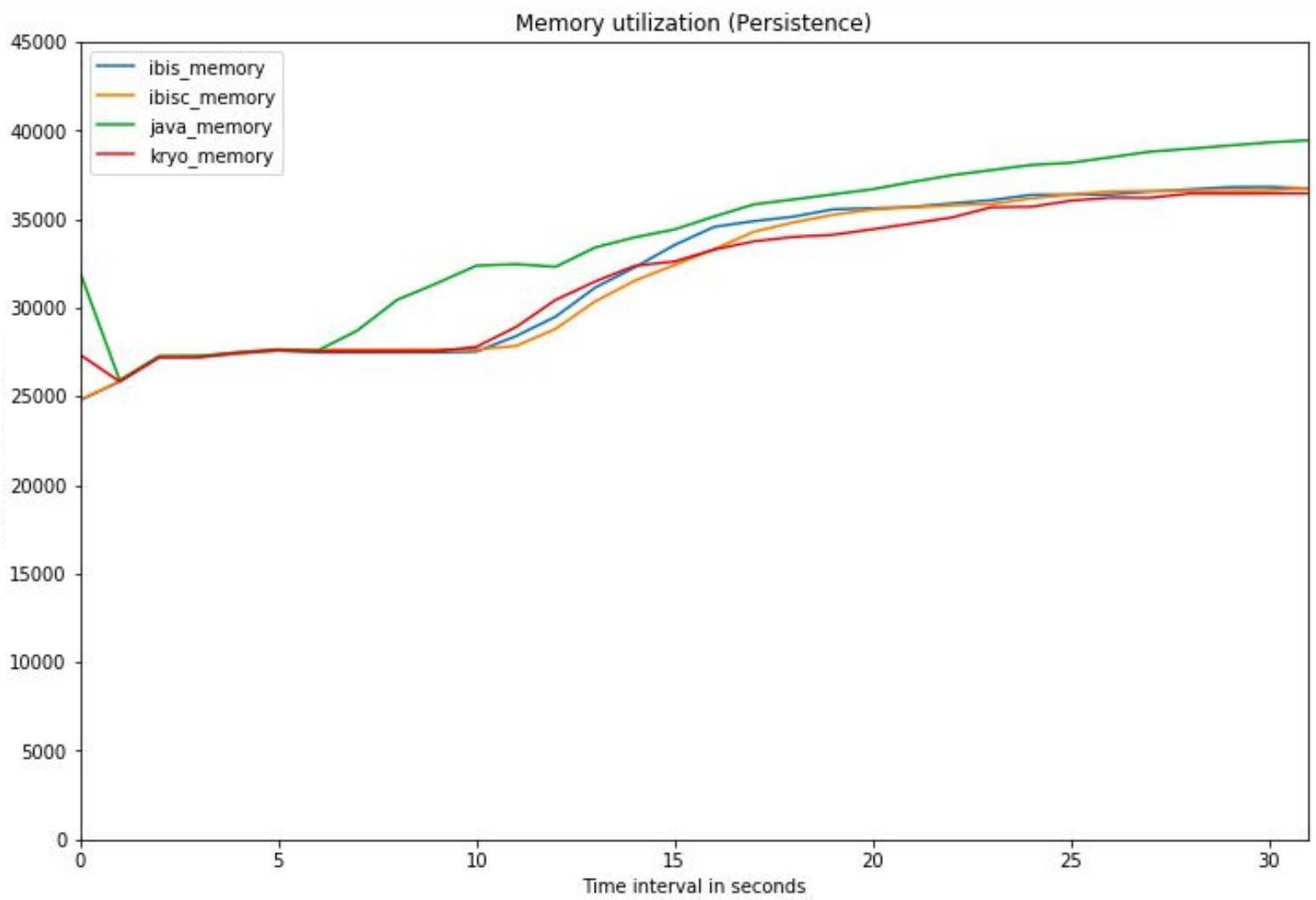
Figure 3: TeraSort time to completion
Time in seconds
0
10
20
30
40
41.43
41.45
34.65
34.57
Java
Kryo
Ibis
Ibisc


Memory Utilization (Terasort)
ibis_memory
ibisc_memory
java_memory
kryo_memory
Memory usage in mbit
Time interval in seconds
0
5000
10000
15000
20000
25000
30000
35000
40000
45000
0
5
10
15
20
25
30


SparkPi time to completion
Time in seconds
15
10
5
0
13.64
13.83
13.8
13.77
Java
Kryo
Ibis
Ibisc


Memory Utilization (SparkPI)
ibis_memory
ibisc_memory
java_memory
kryo_memory
Memory usage in (unit)
Time interval in seconds
0
5000
10000
15000
20000
25000
30000
35000
40000
45000
0
2
4
6
8
10


Persistence time to completion
Time in seconds
0
10
20
30
40
50
50.31
47
42.31
41.7
Java
Kryo
Ibis
Ibisc


Memory utilization (Persistence)
ibis_memory
ibisc_memory
java_memory
kryo_memory
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
0
5
10
15
20
25
30
Time interval in seconds

Conclusion

- Research question:
  - Can Apache Spark's performance be improved by taking advantage of Ibis' serialization techniques?

- 15 out of 17 components could be replaced
- Ibis was 15-20% faster in benchmarks that extensively use serialization
- Ibis was 10-15% more efficient in memory usage in benchmarks that extensively use serialization
- There was no noticeable performance difference in purely computational benchmarks

Future Work

- Replace remaining two components with Ibis serialization
- Measure performance using other benchmarks
- Research performance on a larger scale
- Apply Ibis rewriter to Spark
- Compare Ibis against dataset encoders
- Experiment with Ibis' networking implementations in Spark
- Investigate Ibis serialization performance in other distributed applications

Questions?