### Motivation

- **Inter-node data transfer is a CPU-intensive bottleneck in distributed big data systems**

![Graph showing performance breakdown of Spark's Triangle Counting over the LiveJournal graph](image)

- **Existing approaches**
  - Kryo is faster, but developer must create S/D functions
  - User-defined functions are invoked for every field of every object
  - Correct and efficient S/D functions are labor-intensive to write

We want a systematic solution to reduce S/D costs

### Skyway Approach

**Main Idea:**
transfer *entire* object

- Does not invoke serialization/deserialization functions
- Leads to savings in: 1. CPU time spent on runtime reflection, and 2. Manual developer involvement

Object layout in JVM and how an object is handled by the Java Serializer and Skyway

**Skyway vs. Conventional Data Transfer**

<table>
<thead>
<tr>
<th>Sender</th>
<th>Skyway</th>
<th>Receiver</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Object domain</strong></td>
<td><strong>Serialization</strong></td>
<td><strong>Deserialization</strong></td>
</tr>
<tr>
<td><strong>Binary domain</strong></td>
<td><strong>Network</strong></td>
<td><strong>Binary domain</strong></td>
</tr>
<tr>
<td>Machine dependent meta data, reference, etc.</td>
<td>Machine independent meta data, reference, etc.</td>
<td>Machine independent object data</td>
</tr>
</tbody>
</table>

### Design and Implementation

**Challenges**
- Reference fields
- Type representation

**Solutions**
- Relative addresses with linear time adjustment
- Automated global type numbering system

**Implemented in Oracle’s production JVM OpenJDK 8 build 25.71**
- Across-the-stack modifications: the object/heap layout, the classloader subsystem, the production Parallel Scavenge garbage collector
- Skyway library to interact with runtime systems

### Evaluations

**Java Serializer Benchmark Set**

- An extensive, comprehensive comparison with the existing 90 serializers

![Graph showing Java Serializer Benchmark Set](image)

Skyway outperforms all existing S/D libraries

**Apache Spark 2.1.0 (released December 2016)**

- An 11-node cluster: each has 16 cores, 32GB memory, 1 SSD, connected via InfiniBand
- Four applications on four real-world graphs: LiveJournal, Orkut, UK-2005, and Twitter

Makes Spark run 36% and 16% faster than with Java and Kryo S/D

**Apache Flink 1.3.2 (released August 2017)**

- An 11-node cluster: each has 16 cores, 32GB memory, 1 SSD, connected via InfiniBand
- 5 TPC-H SQL Queries using 100GB of input

Makes Flink run 19% faster than with built-in S/D

### Conclusions

- **Skyway is novel:** the first to provide S/D-free data transfer
- **Skyway is efficient:**
  - Outperforms all existing S/D libraries by 2.2x — 67.3x
  - Improves Apache Spark by up to 73% (Java), and up to 54% (Kryo)
  - Improves Apache Flink by 12% — 29%
- **Skyway is practical:** a JVM-based solution, applicable to all JVM-based languages while requiring zero user effort