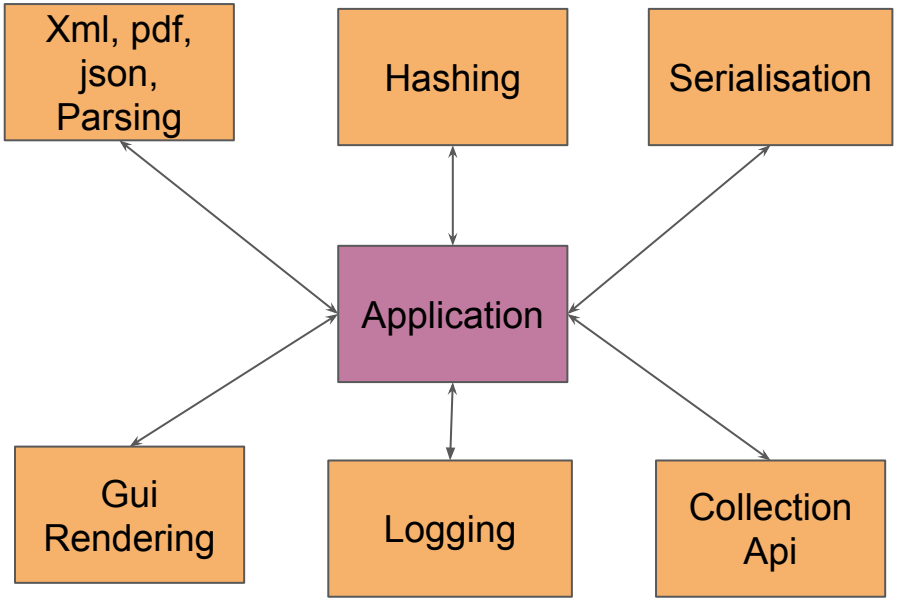
TURINTECH GCARE UNIVERSITY OF SURREY UCL

Checkers: Multi-modal Darwinian API Optimisation

Santanu Kumar Dash (Surrey University), Fan Wu, Michail Basios, Lingbo Li, Leslie Kanthan (Turing Intelligence Technology)

The Problem

Which API/library should we use?



Awesome Java

A curated list of awesome Java frameworks, libraries and software.

GUI

Libraries to create modern graphical user interfaces.

- JavaFX - Successor of Swing.
- Scene Builder - Visual layout tool for JavaFX applications.
- SWT - Graphical widget toolkit.

High Performance

Everything about high-performance computation, from collections to specific libraries.

- Agrona - Data structures and utility methods that are common in high-performance applications.
- Disruptor - Inter-thread messaging library.
- Eclipse Collections - Collections framework inspired by Smalltalk.
- fastutil - Fast and compact type-specific collections.
- HPPC - Primitive collections.
- JCTools - Concurrency tools currently missing from the JDK.
- Koloboke - Hash sets and hash maps.

JSON

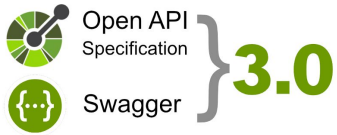
Libraries for serializing and deserializing JSON to and from Java objects.

- DSL-JSON - JSON library with advanced compile time databinding.
- Genson - Powerful and easy-to-use Java-to-JSON conversion library.
- Gson - Serializes objects to JSON and vice versa. Good performance with on-the-fly usage.
- HikariJSON - High-performance JSON parser, 2x faster than Jackson.
- jackson-modules-java8 - Set of Jackson modules for Java 8 datatypes and features.
- Jackson-datatype-money - Open-source Jackson module to support JSON serialization and deserialization of JavaMoney data types.
- Jackson - Similar to GSON, but offers performance gains if you need to instantiate the library more often.
- JSON-io - Convert Java to JSON. Convert JSON to Java. Pretty print JSON. Java JSON serializer.
- jsoniter - Fast and flexible library with iterator and lazy parsing API.
- LoganSquare - JSON parsing and serializing library based on Jackson's streaming API. Outperforms GSON & Jackson's library.
- Moshi - Modern JSON library, less opinionated and uses built-in types like List and Map.
- Yasson - Binding layer between classes and JSON documents similar to JAXB.
- fastjson - Very fast processor with no additional dependencies and full data binding.
- Jolt - JSON to JSON transformation tool.
- JsonPath - Extract data from JSON using XPATH-like syntax.
- JsonSurfer - Streaming JsonPath processor dedicated to processing big and complicated JSON data.

The Problem

Performance impact of APIs

- API selection may affect significantly non functional properties of the code (execution time, memory consumption, energy consumption)
- Microservices architecture and API-first approach can help us achieve automatic API replacement
- APIs deprecated (newer APIs may be faster)
- API tuning (some APIs expose parameters for tuning)



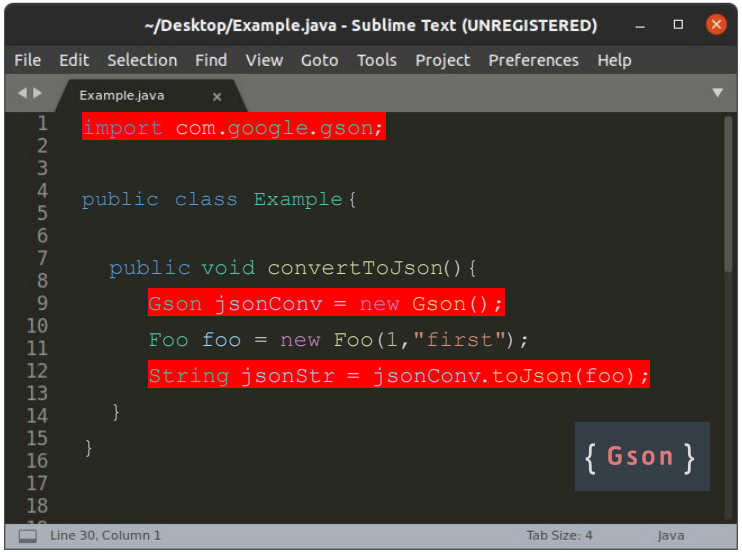
"Amazon says that CodeGuru — which encodes AWS' best practices — has been used internally to optimize 80,000 applications, leading to tens of millions of dollars in savings. In fact, Amazon claims that some teams were able to reduce processor utilization by 325% and lower costs by 39% in just a year."

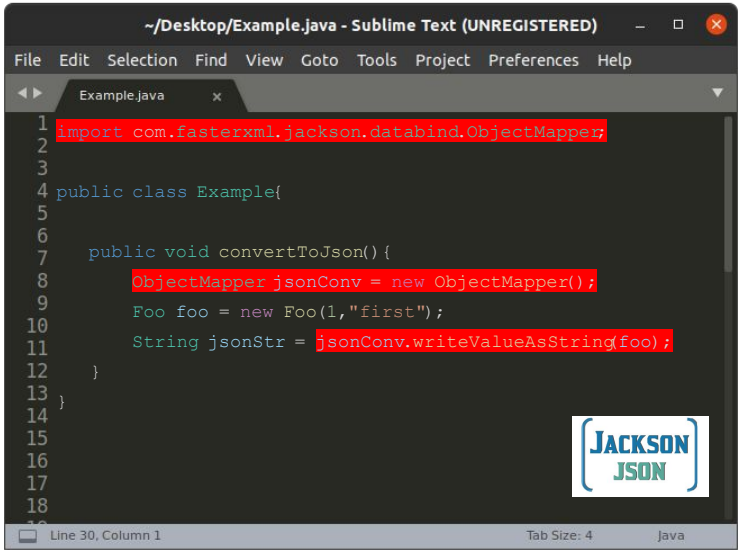
https://venturebeat.com

TURINTECH

The Problem

Example: Json Serialisation in Java





TURINTECH

The Problem

Performance impact of Json library selection

	% of Jackson's parsing speed	% of time increase over Jackson
Jackson	100% (baseline)	0% (baseline)
GSON	47%	111%
JSON.simple	98%	1.3%
JSONP	68%	46.9%

OverOps

Source: https://blog.overops.com/the-ultimate-json-library-json-simple-vs-gson-vs-jackson-vs-json/

TURINTECH

Example: Collection Apis

ArrayList (JCF)

```
List<Integer> integers = new ArrayList<Integer>();
integers.add(1);
integers.add(2);
```

FastList (EC)

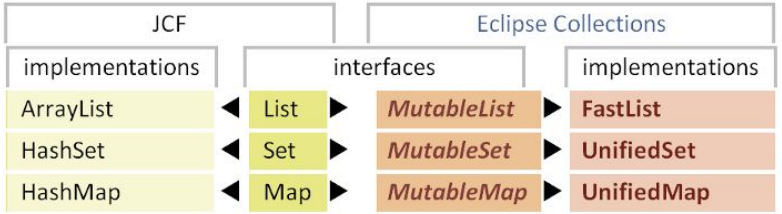
```
List<Integer> integers = new FastList<Integer>();
integers.add(1);
integers.add(2);
```

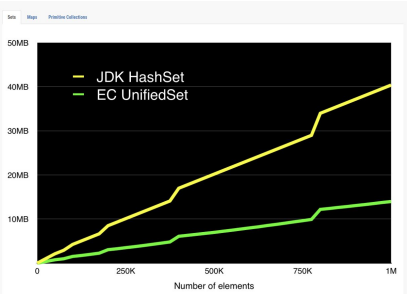
Thread safe

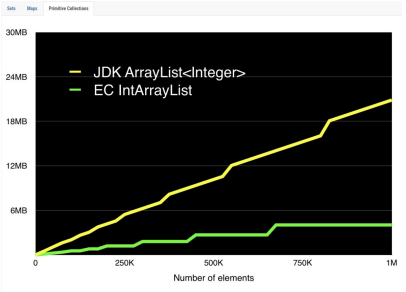
```
ImmutableList<Integer> list =
     Lists.mutable.with(1,2).toImmutable();
```

Memory optimised

```
IntArrayList emptyList = new IntArrayList();
IntArrayList intList = IntArrayList.newListWith(1,2);
```





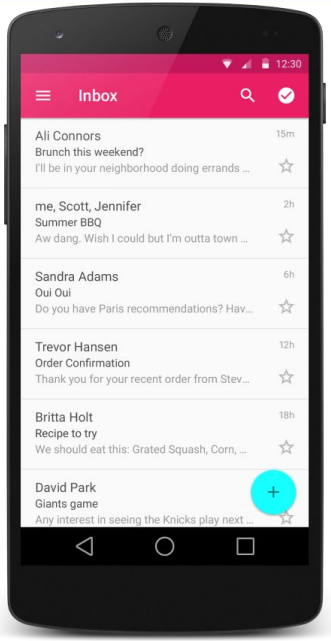


The Problem

Example with API synthesis

- *ListAdapter* API which diffs the lists on a background thread unblocking the main thread
- *AsyncListDiffer* which does the same task through a callback
- Low-level *DiffUtil* class which achieves the same task on a background thread

Each of the three techniques use a combination of API calls but are semantically equivalent.



RecyclerView

In the RecyclerView model, several different components work together to display your data.

TURINTECH

Searching for equivalent APIs

Definition 2.1. The *type environment* Γ for a program is the mapping of terms in a grammar to their type where $\Gamma(x)$ returns the type for the terms x. If x is a method, $\Gamma^i(x)$ returns a sequence of its input types and $\Gamma^o(x)$ returns a sequence of its output types.

For every API call site, there are two ways in which potential replacements can be identified: a) **Singular** and **Compositional** replacement

Definition 2.2. *Singular Replacement* is the replacement of a single API call f with g such that $\Gamma(f) = \Gamma(g)$

Definition 2.3. *Compositional Replacement* is the replacement of a single API call f with a sequence of type-correct API calls $g_1(g_2(g_3(\cdots g_n(x_1, x_2, \cdots, x_k)\cdots)))$ such that $\forall k.\Gamma^i(g_k) = \Gamma^o(g_{k+1}) \wedge \Gamma^i(g_n) = \Gamma^i(f) \wedge \Gamma^o(g_1) = \Gamma^o(f)$

TURINTECH

Proposed Framework

Multi Stage API Optimisation

- **Identify** stage parses the source to identify locations for target APIs
- **Transform** stage searches for candidate replacement amongst API models
- **Test** phase runs unit and integration tests on the rewritten code to sanity check the rewriting

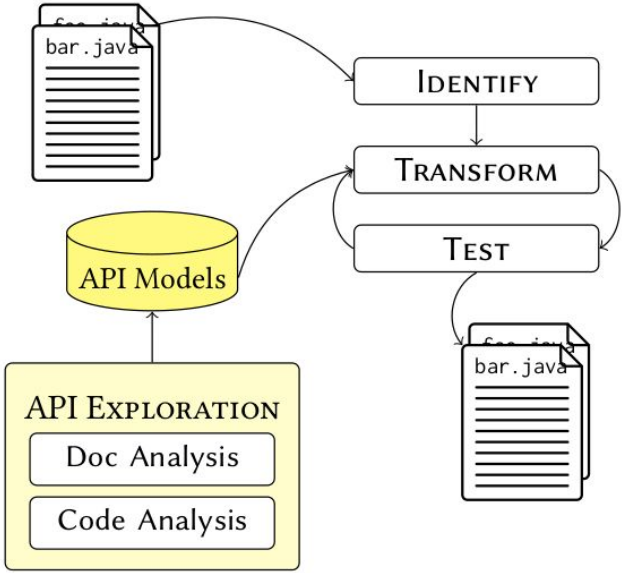
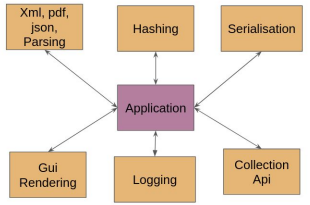


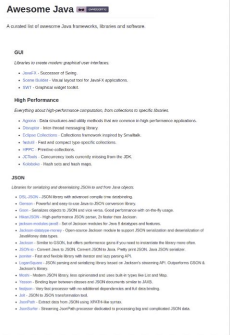
Figure 1: Overview of CHECKERS.

TURINTECH

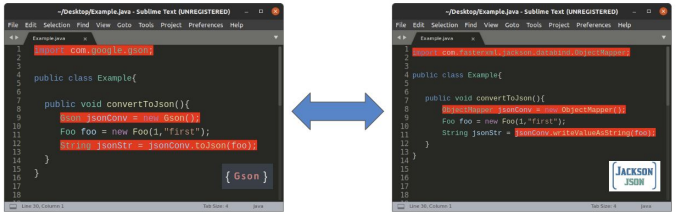
Questions?

Which API/library should we use?





Example: Json Serialisation in Java



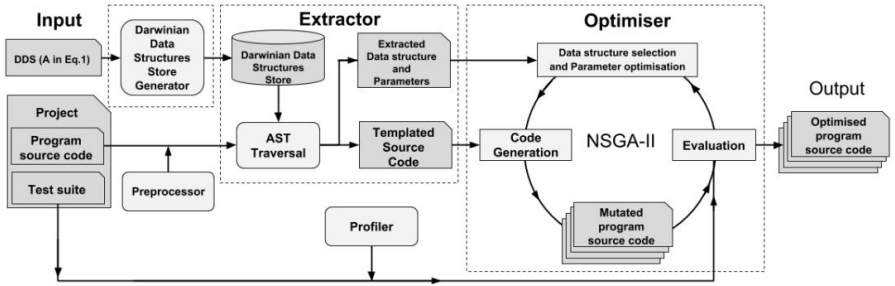


Figure 1: System Architecture of ARTEMIS.

mike@turintech.ai

TURINTECH