Performance Benchmarking of FHIR Terminology Operations in ETL Jobs

Joshua WIEDEKOPF, Cora DRENKHAHN, Josef INGENERF
Institute of Medical Informatics
University of Lübeck, Germany

Medical Informatics Europe 2023, Gothenburg, 🇸🇪
Motivation

- National initiatives demand data from Healthcare providers for selected use cases
- ETL jobs perform data mapping for these initiatives into standardized formats
  - Syntactic mapping: transfer a datum into a standardized data structure (e.g. from relational data, HL7 v2, … to HL7 FHIR profile, openEHR template, …)
  - Semantic mappings
    - Unit conversions (e.g. from internal unit codes to UCUM)
    - NLP and other AI methods
    - Mapping of coded data (e.g. from internal laboratory codes to LOINC)
HL7 FHIR Terminology Module

- Groundwork for terminology servers
- Framework has seen broad use, even outside of HL7 FHIR
- Definition of resources for terminological content
- Definition of operations between a client system and a server/system providing these resources
  - *Is a code a member of a CodeSystem/ValueSet?* (CS/$validate-code)
  - *What is the definition of a code in this CodeSystem?* (CS/$lookup)
  - *Map from this code to this code using this ConceptMap (CS/$translate)*
  - …

Image: [https://www.hl7.org/fhir/terminology-module.html](https://www.hl7.org/fhir/terminology-module.html) [simplified]
Benchmarking FHIR Terminology

- Use Case: ETL job for a partner in the *Use Case Infection Control* of the *HiGHmed* consortium of the MI-I
- We have access to a high-performance terminology server (locally), but...
  - Dependency on server during all phases of development and deployment
  - Performance impact/bottleneck of continuous HTTP requests to the server?
  - Importance of caching?
Benchmarking FHIR Terminology

• Use Case: ETL job for a partner in the Use Case Infection Control of the HiGHmed consortium of the MI-I

• We have access to a high-performance terminology server (locally), but…
  – Dependency on server during all phases of development and deployment
  – Performance impact/bottleneck of continuous HTTP requests to the server?
  – Importance of caching?

How do I integrate terminology services into my ETL job without incurring a significant bottleneck?
Benchmarking setup

- measurement of $ops/s$ for two operations
  - `CodeSystem/$lookup`
  - `ConceptMap/$translate`
- Multiple implementations of the same functionality benchmarked one after the other
- Several different approaches to caching
- Generation of test dataset from real-world resources
  - $ops/s_{input} \gg ops/s_{operations}$

https://github.com/openjdk/jmh
Benchmarking setup

- measurement of $ops/s$ for two operations
  - CodeSystem/$lookup$
  - ConceptMap/$translate$
- Multiple implementations of the same functionality benchmarked one after the other
- Several different approaches to caching
- Generation of test dataset from real-world resources
  - $ops/s_{input} \gg ops/s_{operations}$

- JMH: Java Microbenchmarking Harness

https://github.com/openjdk/jmh
Benchmarking setup

- measurement of $ops/s$ for two operations
  - `CodeSystem/$lookup`
  - `ConceptMap/$translate`
- Multiple implementations of the same functionality benchmarked one after the other
- Several different approaches to caching
- Generation of test dataset from real-world resources
  - $ops/s_{input} \gg ops/s_{operations}$

- JMH: *Java Microbenchmarking Harness*
- Industry-standard setup for generating reliable benchmarks

https://github.com/openjdk/jmh
Benchmarking setup

- measurement of $ops/s$ for two operations
  - CodeSystem/$lookup$
  - ConceptMap/$translate$
- Multiple implementations of the same functionality benchmarked one after the other
- Several different approaches to caching
- Generation of test dataset from real-world resources
  - $ops/s_{input}$ $\gg$ $ops/s_{operations}$

- JMH: Java Microbenchmarking Harness
- Industry-standard setup for generating reliable benchmarks
- JMH greatly simplifies benchmarking setup, but *caveat emptor!*

https://github.com/openjdk/jmh
Implementations

**File**

- Setup: do nothing
- Bench: open file
- Service: linear scan

**RAM HashMap**

- Setup: load CodeSystems into HashMap
- Bench: load ConceptMaps into HashMap
- Service: get from HashMap
Implementations

Redis / SQLite

HTTP

Icons: Material Design Icons via https://pictogrammers.github.io/@mdi/font/6.9.96
Performance Benchmarking of FHIR Terminology Operations

Results

<table>
<thead>
<tr>
<th>Input</th>
<th>File</th>
<th>RAM</th>
<th>SQLite</th>
<th>Redis</th>
<th>HTTP</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>None</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Redis</td>
<td>1.6e3</td>
<td>1.0e3</td>
<td>2.8e4</td>
<td>2.8e4</td>
<td>2.8e4</td>
</tr>
<tr>
<td>HashMap</td>
<td>2.0e4</td>
<td>2.2e4</td>
<td>3.3e6</td>
<td>2.2e4</td>
<td>2.0e5</td>
</tr>
<tr>
<td>JSON-HashMap</td>
<td>2.0e4</td>
<td>2.2e4</td>
<td>3.3e6</td>
<td>2.2e4</td>
<td>2.0e5</td>
</tr>
<tr>
<td>Caffeine</td>
<td>None</td>
<td>Caffeine</td>
<td>None</td>
<td>None</td>
<td>Server-side</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Ktor</td>
</tr>
</tbody>
</table>

log10ops/s
Results
Results

- HTTP is the slowest, while RAM is the fastest
Results

- HTTP is the slowest, while RAM is the fastest
- Caching is very beneficial for all operations
  - Implementation of the cache is not as important
Results

- HTTP is the slowest, while RAM is the fastest
- Caching is very beneficial for all operations
  - Implementation of the cache is not as important
- Network operations may hurt performance (Redis vs SQLite)
• HTTP is the slowest, while RAM is the fastest

• Caching is very beneficial for all operations
  – Implementation of the cache is not as important

• Network operations may hurt performance (Redis vs SQLite)

• `$translate` is more involved than `$lookup`, but pattern is similar
Discussion

- Obvious limitations of this study
- Caching is important!
- Local solutions will often perform better than querying a FHIR TS across the Internet
  - Need for national and supranational provision of relevant resources for local deployment
  - Rolling your own solution is not trivial!
- Requirements and circumstances of the individual deployment must be taken into account when incorporating local terminology operations
  - Maybe even perform your own benchmarks on your own hardware
  - Consider hybrid approaches: delegate complex operations and implement simple ops yourself
Contact

Joshua Wiedekopf, M.Sc.
Research Associate

University of Lübeck
IT Center for Clinical Research &
Institut für Medizinische Informatik
Ratzeburger Allee 160
23562 Lübeck

- j.wiedekopf@uni-luebeck.de
- @jpwiedekopf
- www.linkedin.com/in/jpwiedekopf