Análisis, mejora del código y pruebas de la librería CF4J: Collaborative Filtering for Java

Autor: Dionisio Cortés Fernández
Tutor: Francisco Gortázar Bellas

Objetivos

- Mejorar un proyecto existente
- Aplicar técnicas de CI
- Aplicar técnicas de mutation testing
- Aplicar técnicas de carga de forma local y en cloud

¿Que es?

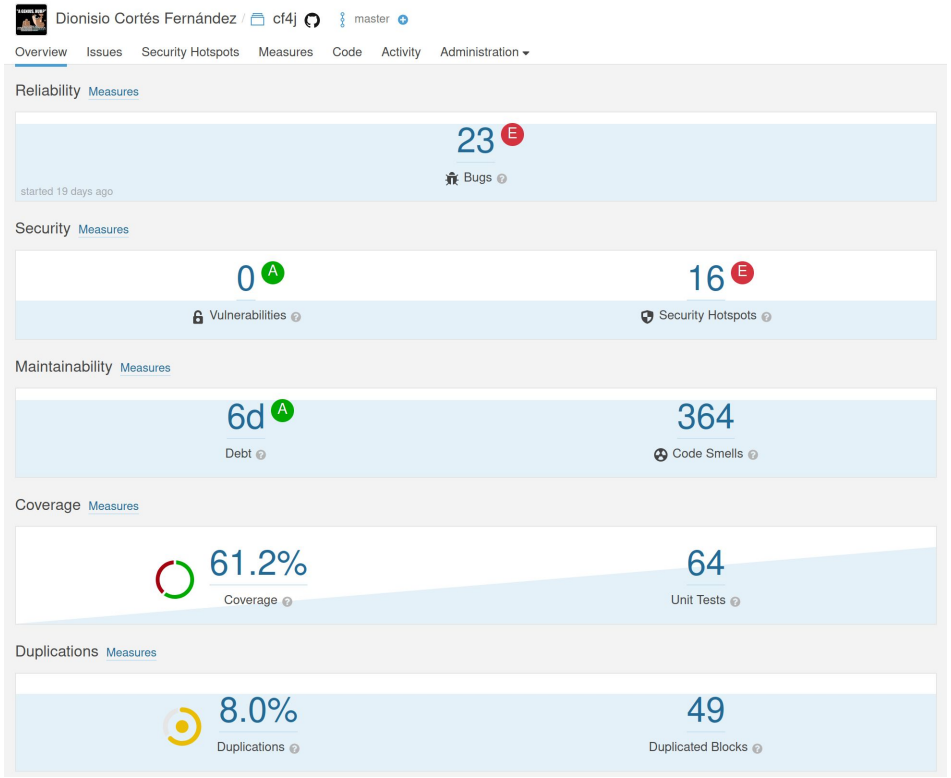
cf4j es una librería que implementa diversos algoritmos de sistemas de recomendación y que es muy útil para gente que quiere adentrarse en ese mundo y una referencia en el mundo académico.

Pasos

- Crear pipeline con github actions
- Añadir SonarQube para ver el estado
- Aplicar mutation testing
- Crear una aplicación rest para usar la libreria
- Load testing

Primera ejecución

- 23 Bugs
- 16 Security hotspots
- 61.2% Coverage
- 8% Duplications



Mutation testing

- Verificamos el estado de los tests
- Bastante costoso computacionalmente
- Nuevas técnicas como extreme mutation testing
- Muy estudiado con técnicas de machine learning

Mutation testing comparativa

	Mutation testing	Extreme mutation testing
Line coverage	66%	67%
Mutation coverage	61%	58%
Test strength	83%	94%

Mutation comparativa tiempos

Timings	Mutation testing	Extreme mutation testing
pre-scan for mutations	< 1 second	< 1 second
scan classpath	< 1 second	< 1 second
coverage and dependency analysis	2 seconds	2 seconds
build mutation tests	< 1 second	< 1 second
run mutation analysis	2 minutes and 53 seconds	1 minutes and 8 seconds
Total	2 minutes and 57 seconds	1 minutes and 11 seconds
Ran tests	2275	273
Total time	06:02 min	02:32 min

Implicaciones de las métricas de calidad

- Muy necesarias para ver el estado del proyecto
- Necesitamos que todo el mundo sea consciente de ellas
- Integradas lo antes posible en el ciclo de desarrollo

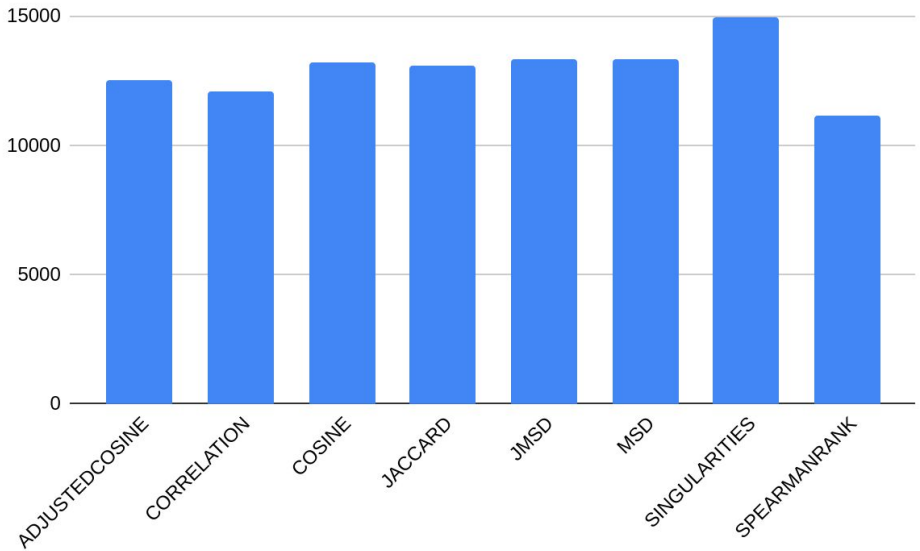
Evolución de las métricas de calidad

Metric	I1	I2	I3	I4	M1	M2	M3
code_smells	364	359	358	360	344	344	360
bugs	23	21	19	5	7	7	7
duplicated_lines	902	902	902	935	943	943	1125
duplicated_lines_density	8.0	8.1	8.1	8.4	7.9	7.9	8.7
violations	387	380	377	365	351	351	367
coverage	61.2	62.1	62.1	62.1	63.5	63.5	59.7
security_hotspots	16	16	16	5	7	5	7

Pruebas de carga

- Creación de una aplicación rest que usa la librería, java 17 y springboot 2.6.1
- Aplicación con CI/CD y smoke tests para ver que todo funciona

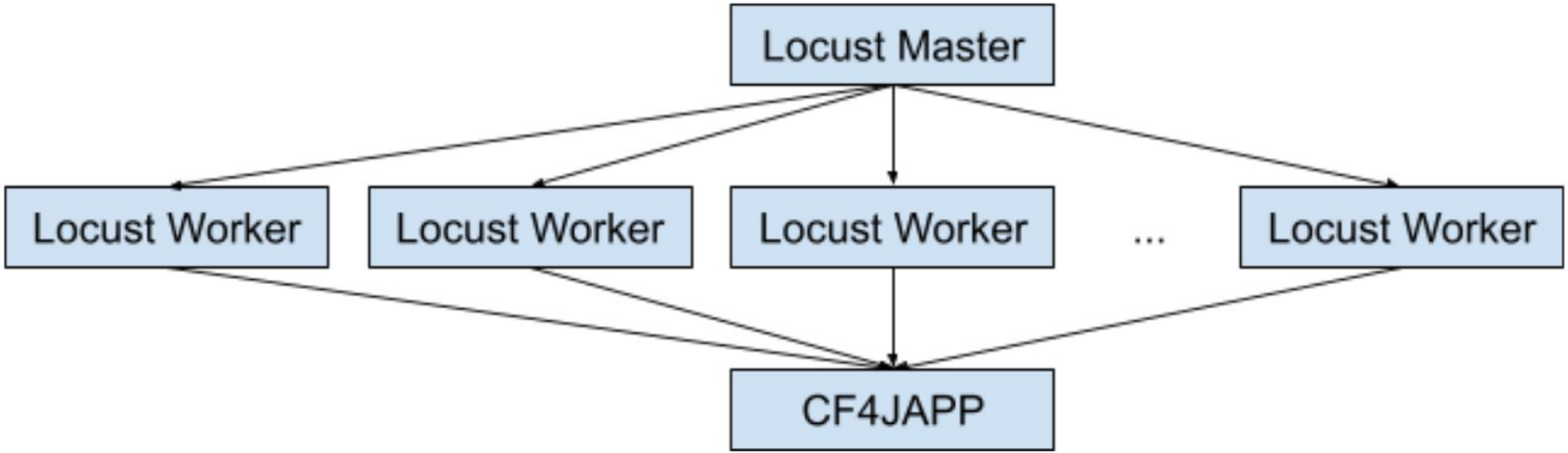
Pruebas de carga Jmeter



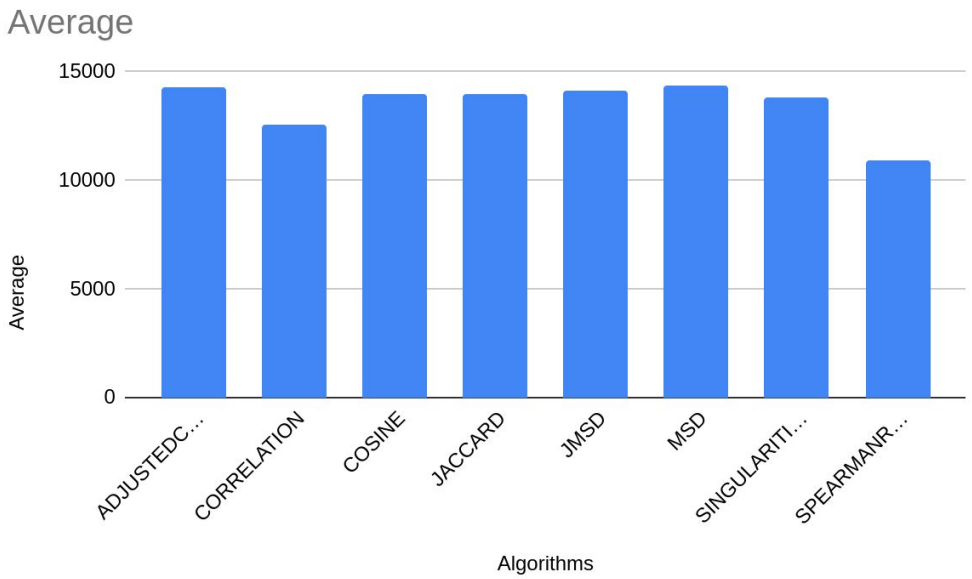
Pruebas de carga cloud

- Creación de un cluster en google cloud
- Despliegue de yamls de kubernetes
- Evolución a helm

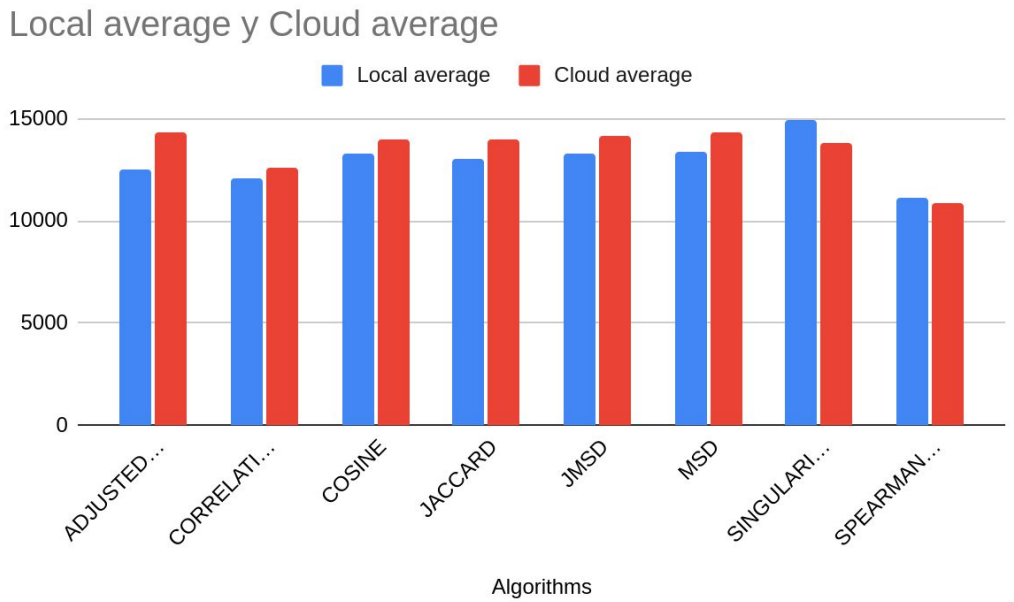
Pruebas de carga cloud



Pruebas de carga cloud



Pruebas de carga comparativa



Conclusiones

- Mayor dificultad proyectos colaborativos
- Importancia métricas de calidad y que sean automáticas
- Mutation testing costoso pero útil
- Importancia de las pruebas en un entorno real

Pruebas de carga cloud

- Profundizar en qué operadores son necesarios para mejorar la hipótesis del programador competente.
- Analizar código de github y usar técnicas de machine learning para hacer tests de mutación automáticos en base a ese aprendizaje.
- Profundizar en cómo se podría mejorar la librería desde un punto de vista de la arquitectura del software.
- Migrar travisCI a github actions.
- Automatizar la creación del entorno.
- Poner sonar en el repositorio.