МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ
ПО НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

Тема: Fuzzy планирование потоков с применением lincheck для обнаружения ошибок в многопоточном коде

Студент гр. 5304 _____ Шахов А.Д.

Преподаватель _____ Калишенко Е.Л.

Санкт-Петербург
2020

ЗАДАНИЕ

НА НАУЧНО-ИССЛЕДОВАТЕЛЬСКУЮ РАБОТУ

Студент Шахов А.Д.

Группа 5304

Тема работы: Fuzzy планирование потоков с применением lincheck для обнаружения ошибок в многопоточном коде

Задание на НИР:
Разобраться со способами внедрения через обёртку над Atomic*Reference с барьерами и соответствующий ClassLoader

Сроки выполнения НИР: 01.09.2020-15.12.2020
Дата сдачи отчета: 15.12.2020
Дата защиты отчета: 15.12.2020

Студент гр. 5304 _____ Шахов А.Д.

Преподаватель _____ Калишенко Е.Л.

АННОТАЦИЯ

Целью работы является внедрение через обёртку над Atomic*Reference поведения управлением fyzzy-планированием потоков. В данной работе представлены основные подходы к реализации внедрения нового поведения в уже существующие классы. Представлен результаты внедрения измененных методов наиболее перспективными способами. Выбрано направление дальнейшего развития.

SUMMARY

The aim of the work is implementation through the Atomic * wrapper. Management of fyzzy-scheduling of threads. In this work, the main implementation approaches to the class of the new behavior in the already correct one. The results of the introduction of the modified methods by the most promising methods are presented. Further development is chosen.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Текущие проверки линеаризуемости алгоритмов, например с помощью lincheck, заключаются в случайном многопоточном исполнении кода и сравнении результатов со всеми возможными последовательными исполнениями тех же операций[2]. В ходе проверки даже самых простых студенческих lock-free алгоритмов выявилась полезность следующих функций: останавливать потоки после / перед CAS для повышения выроятностью некорректных ситуаций;
долгой остановки потока после / перед CAS с проверкой отработки остальных потоков по своим задачам.

1. ПОСТАНОВКА ЗАДАЧИ

1.1. Актуальность

Каждый, кто хоть раз писал многопоточный код, понимает, насколько легко в нем допустить ошибку. Более того, в сложных алгоритмах некоторые ошибки воспроизводятся крайне редко и только на определенных исполнениях, а, значит, простыми тестами их обнаружить проблематично. Поэтому так важно совершенствовать существующие подходы к решению данной проблемы.

1.2. Проблема

Текущие проверки линеаризуемости алгоритмов, например с помощью lincheck, заключаются в случайном многопоточном исполнении кода и сравнении результатов со всеми возможными последовательными исполнениями тех же операций. В ходе проверки даже самых простых студенческих lock-free алгоритмов выявилась полезность следующих функций:

- Останавливать потоки после / перед CAS для повышения вероятностью некорректных ситуаций
- Долгой остановки потока после / перед CAS с проверкой отработки остальных потоков по своим задачам

1.3. Цель

Внедриться в JRE или байт-код и на основе понимания использования CAS[3] явно влиять на планирование потоков, чтобы расширить количество классов ошибок, выявляемых с помощью lincheck.

1.4. Задачи

- Разобраться со способами внедрения через обёртку над Atomic*Reference с барьерами и соответствующий ClassLoader
- Внедрение обёртки при помощи Java instrument

2. РЕЗУЛЬТАТЫ РАБОТЫ В ОСЕННЕМ СЕМЕСТРЕ

2.1. Разобраться со способами внедрения через обёртку над Atomic*Reference с барьерами и соответствующий ClassLoader

Для засыпания операций, необходимо внести изменение в стандартные классы, которые использует программа. Существует два варианта: подмена ClassLoader`a, либо использование Java instrument.

Главным плюсом подмены ClassLoader`a является простота реализации, можно создать свой собственный, кастомный ClassLoader, либо воспользоваться готовыми библиотеками. Однако у данного подхода есть существенные недостатки, а именно: управление возможно только через параметры jvm т.е. на старте программы; возможна только статическая инициализация; подмененный класс будет использоваться во всей программе, что может сильно замедлить саму проверку линеаризуемости алгоритма.

Использование Java instrument, позволяет динамическую замену байт-кода, это возможно даже в конкретном потоке. Также такой подход является классическим в задачах подобного рода[4] и широко используется, например, компанией jRebel. Главным минусом является большое количество дополнительных действий, вызванных использованием сложной цепочки для подмены класса.

2.2. Внедрение обёртки при помощи Java instrument

Реализацию данного механизма можно разделить на несколько шагов:

1. При помощи Javassist генерируем код, который будет реализовывать дополнительное поведение.
2. При помощи Java Agent изменяем код существующего класса, добавляя ему поведение реализованное на предыдущем шаге.
3. При помощи Attach API производим эту замену во время работы JVM без предварительной установки параметров её запуска.

2.3. Код программы

https://github.com/sh-ad/lock-free-testing

3. ПЛАН РАБОТЫ НА ОСЕННИЙ СЕМЕСТР

На весенний семестр запланировано:

- Реализовать при помощи Attach API внедрение обёрток над классами
- Исследование изменения качества проверки многопоточного кода
- Написание ВКР

ЗАКЛЮЧЕНИЕ

В ходе выполнения научно-исследовательской работе была исследована возможность применение fuzzy-планирования потоков при тестировании многопоточных алгоритмов. Теоретические предположения были проверены на практике. Было реализовано детектирование ошибки нарушения lock-free контракта на примере с Set при отсутствии физического удаления во время добавления элементов. Реализовано наиболее перспективное направление использования fuzzy-планирования потоков.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Духанов, А. В. Системное и прикладное программное обеспечение: конспект лекций / С.И. Абрахин, А.В. Духанов; Владим. гос. ун-т. – Владимир: Изд-во Владим. гос. ун-та, 2010. – 47 с.
2. В.М. Илюшечкин. Операционные системы. – М.: Бином. Лаборатория знаний, 2009. – 112 с. ISBN 978-5-94774-963-2
3. Vimal K., Trivedi A. A memory management scheme for enhancing performance of applications on Android //Intelligent Computational Systems (RAICS), 2015 IEEE Recent Advances in. – IEEE, 2015. – С. 162-166.
4. Hahn S. S. et al. Fasttrack: foreground app-aware I/O management for improving user experience of android smartphones //2018 {USENIX} Annual Technical Conference ({USENIX}{ATC} 18). – 2018. – С. 15-28.
5. Hussein A. et al. One process to reap them all: Garbage collection as-a-service //ACM SIGPLAN Notices. – ACM, 2017. – Т. 52. – №. 7. – С. 171-186.