Автоматическое fuzzy-планирование потоков с помощью relacy для обнаружения ошибок в многопоточном коде

Проблема

Проверка многопоточного кода (особенно lock-free) одной из основных составляющих включает себя постоянное стресс-тестирование, причём не от хорошей жизни. Даже при реализации подобных алгоритмов на Java, так как там есть AtomicReference и нет ABA, никто не отменяет наличия других ошибок || программирования. Текущие проверки обычно сводятся к:

1. проверить производительность (заданная масштабируемость на 1 и N потоках-работниках по идее должна быть сильно выше lock-based версии, если подобрать верно контейнер)
2. отсутствие использования примитивов типа mutex и т.п. (и то иногда бывает полезно задействовать spin-lock)
3. искусственно поостанавливать потоки так, чтобы остальные продолжали выполнять задачи (это уже не просто)
4. ThreadSanitizer или аналоги

Пути решения

С другой стороны, был очень хороший доклад Димы Вьюкова про Fuzzing Test на конференции C++ Russia. Суть в том, что с помощью специальной инструментации кода fuzzer может строить последовательности входных данных “с нуля” такие, которые максимизируют покрытие кода. За 15 минут простейший fuzzer длиной 5 строк кода нашел более 10 ошибок в boost.regexp (boost 1.63). Максим Хижинский немного поговорил с докладчиком, как приспособить этот подход к тестированию lock-free алгоритмов. Сошлись на том, что здесь в качестве “входной последовательности” должен выступать свой собственный планировщик (за отправную точку можно взять relacy), который мог бы планировать потоки так, чтобы максимизировать покрытие кода. Это как-то перекликается с “искусственно поостанавливать потоки” из п.3, но более разумно, так как такой планировщик может заглянуть в instrumented code и посмотреть, что и где нужно останавливать/ запускать и с какими данными

Ближайшие задачи

1. Разобраться на тестовых примерах как работать с relacy
2. Разобраться с LLVM/clang для формирования видения порядка управления потоками во время исполнения
3. Сделать/взять простую реализацию списка и искусственно внести туда ошибки

4. Натравить relacy
5. Продумать шаги, как сделать из relacy что-то типа fuzzer'a

Идеи на обсуждение

Как можно использовать relacy 6 llvm для поиска ошибок при работе с барьерами памяти:

1. Имитировать interleave потоков: искуственно останавливать / запускать потоки около барьеров для имитации наиболее нежелательных ситуаций
2. Имитировать memory reordering: переставлять обращения к памяти имитируюя наиболее weak архитектуры и прогонять код на имеющейся тестовой базе
3. Отслеживание несогласованности применёных в разных потоках (или вообще не применённых) барьеров за счёт анализа типов операций (load/store) над атомиками или переменными вокруг явных барьеров