УДК 004.052.32

РАЗРАБОТКА МЕТОДОВ ВЕРИФИКАЦИИ КОРРЕКТНОСТИ RPC ПРОТОКОЛА

Черкашин Н.К. (ИТМО)
Научный руководитель – PhD, науки, доцент Аксенов В.Е.
(ИТМО)

Введение. В разработке распределенных систем зачастую необходимо вызывать функцию на другой машине. Для этого используется протокол RPC (Remote Procedure Call), который построен поверх других сетевых протоколов - основное различии в их реализации заключается в использовании различных протоколах транспортного уровня, в зависимости от строгости нужных гарантий используют так UDP, так и TCP.

Однако все эти протоколы по природе своей являются недетерминированными, из-за наличия сетевых задержек и потерь трафика, а также потенциальных отказов клиентов. С поддержкой новых гарантий в протоколе возрастает вероятность того, что обычные тесты, даже несмотря на полное покрытие по строкам, не рассматривают все случаи которые могут произойти при использовании протокола. Кроме этого все эти тесты работают недетерминированно, что усложняет нахождение ошибок в исходном коде и их последующее исправление.

Эту проблему можно решить, если использовать системы на основе model checking [1], которые проверяют все исполнения при заданных ограничениях. Однако в большинстве случаев для использования этого подхода надо переписать код в модель на специализированном языке таком как TLA+. Тем самым мы проверяем корректность не исходного кода, а только лишь модели. В свою очередь, наше решение проверяет и дает гарантию именно про исходный код.

Основная часть. Для верификации протокола дорабатывался и использовался фреймворк LTest [2], который проверяет все возможные варианты исполнения исходного кода на языке C++. Этот фреймворк был изначально разработан для проверки многопоточных исполнений, где недетерминированным является поведение: какой поток когда исполняется. В случае RPC-протоколов у нас дополнительно присутствует недетерминизм того в каком порядке и какое сообщение пришло.

Для того, чтобы добиться детерминизма и воспроизводимости исполнений, многопоточное исполнение в фреймворке тестировалось в однопоточной среде, где вместо физических потоков исполнение происходило на виртуальных потоках. Этими потоками управляет планировщик который выбирает какой из виртуальных потоков будет исполняться на следующем шаге, который заканчивается как только появляется точка переключения или конец функции.

Также планировщик генерирует при отсутствии исполняемой задачи на потоке следующую задачу и аргументы к ней.

Вставка точек переключения между виртуальными потоками происходит посредством LLVM pass[3], они добавляются в места которые могут не быть потокобезопасными или где происходит получение сообщений.

Таким образом через рассмотрение всех возможных чередований потоков, рассматриваются все возможные порядки получения сообщений и сбоев при заданных ограничениях.

Были рассмотрены две различные имплементации с протоколами на транспортном уровне UDP и TCP соответственно и проведена доработка инструмента LTest для поддержки их тестирования. Одним из важных изменений была поддержка динамического числа потоков, для тестирования реализации построенной на протоколе TCP. В этом протоколе сервер создает отдельный виртуальный поток под каждого клиента и необходимо верифицировать в том числе факт, что поток для обслуживания одного клиента никогда не может получить данные для другого клиента.

Кроме этого было уменьшено число исполнений, которые необходимо перебрать для проверки всех возможных состояний системы. Исполнения приводят к одинаковому состоянию системы, если их операции на потоках эквиваленты друг другу по модулю перестановок потоков и наличия операций, которые можно переставить между потоками и не нарушить историю. Рассмотрев лишь одно исполнение из каждого класса эквивалентности, полученные под таким отношением, мы переберем все возможные состояния системы.

Выводы. Был адаптирован фреймворк LTest под поддержку тестирования RPC протокола и написаны тесты которые повысили уверенность в корректной работе протокола.

Список использованных источников:

1. Lamport L. Specifying systems: the TLA+ language and tools for hardware and software engineers. – 2002.
2. LTest [Электронный ресурс]. — URL: https://github.com/ITMO-PTDC-Team/LTest(дата обращения 08.02.2025)
3. Writing a LLVM Pass [Электронный ресурс]. — URL: https://llvm.org/docs/WritingAnLLVMNewPMPass.html (дата обращения: 13.02.2025).