УДК 004.62

ПРИМЕНЕНИЕ ТЕХНОЛОГИИ БЛОКЧЕЙН ДЛЯ ПОВЫШЕНИЯ НАДЕЖНОСТИ МУЛЬТИВЕРСИОННОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Грузенкин Д.В., Михалев А.С., Царев Р.Ю., Суханова А.В., Новиков О.С.
ФГАОУ ВО «Сибирский федеральный университет», Красноярск, e-mail: gruzenkin.denis@good-look.su

На сегодняшний день надёжность программного обеспечения в некоторых сферах науки и техники критически важна. К таким сферам относятся авиация, космические исследования, химическая промышленность и пр. Одним из хорошо себя зарекомендовавших подходов к повышению надёжности программного обеспечения является применение методологии мультиверсионного программирования. Мультиверсионное программное обеспечение предполагает реализацию ряда версий его модулей, различных по определённому признаку, но выполняющих одну и ту же задачу, что обеспечивает независимость потенциальных сбоев. Однако в силу архитектурных особенностей аппаратной реализации систем управления могут возникать зависимости между версиями или же модулями. Это приводит и к зависимости между возможными сбоями. Для их выявления, а также для повышения целостности данных, описывающих работу системы (логов), предлагается применение технологии блокчейн в качестве средства логирования. В статье описан численный эксперимент, который проводился для проверки истинности гипотезы о повышении уровня надёжности мультиверсионного ПО вследствие более детального и надёжного учета и анализа системных логов. Поставленный эксперимент подтвердил выдвинутую гипотезу – уровень надёжности мультиверсионной программной системы повышается при использовании более полной информации о работе системы. Более полная информация о работе системы, представленная в виде большего количества логов, достигается путём применения технологии блокчейн.

Ключевые слова: мультиверсионное программное обеспечение, блокчейн, надёжность программного обеспечения, логирование

APPLICATION OF THE BLOCKCHAIN TECHNOLOGY TO INCREASE N-VERSION SOFTWARE RELIABILITY

Gruzenkin D.V., Mikhalev A.S., Tsarev R.Yu., Sukhanova A.V., Novikov O.S.
Siberian Federal University, Krasnoyarsk, e-mail: gruzenkin.denis@good-look.su

Nowadays software reliability is crucial in such fields as aviation, space research, the chemical industry, and so on. N-version programming is one of the approaches to ensure a high level of software reliability and its tolerance to faults while executing. N-version software includes a set of versions of its modules. All versions of a module are different but functionally equivalent. The versions are designed according to the same specification. The diversity of the versions tends to avoid interdependency of residual faults in the versions. However, it appears that particularity of hardware architecture leads to interdependencies between versions or even modules. As a result, dependencies between faults arise. To increase software reliability and detect residual faults we propose to apply the blockchain technology to log the software execution process. The article presents results of an experiment of software reliability increasing due to detail and profound analysis of the software logs. The experiment demonstrates that complete information on software execution allows to increase the software reliability. To make the information more complete we have used blockchain technology for detailed logging.

Keywords: N-version software, blockchain, software reliability, logging

В IT-индустрии одни разработки с течением времени изживают себя, теряют актуальность, другие – напротив, появляются и развиваются. Одним из последних этапов развития информационных технологий можно считать технологии, обобщенные названием «блокчейн» (blockchain). В настоящее время их изучению уделяется большое внимание, поскольку блокчейн-технологии можно считать революционным прорывом в сфере информационной безопасности и распределённой обработки данных [1].

Блокчейн – это многофункциональная и многоуровневая информационная технология, предназначенная для надежного учета различных активов (как материальных, так и нематериальных) [2–4]. Информация об активах и операциях над ними шифруется, преобразуясь в так называемый блок [5]. Каждому активу соответствуют свои приватный и публичный ключи, где публичный ключ необходим для осуществления операций над активом, а приватный – для проверки валидности проведения той или иной операции [5]. К числу таких операций относятся, например, финансовые транзакции.

При соответствии публичного ключа приватному ключу транзакция считается одобренной. Сами же ключи индивидуальны для каждого актива и представляют собой генерируемые по определённому алгоритму наборы символов. Чтобы закрыть блок (зафиксировать какое-либо событие), пользователь подбирает ключ, который име-

ет отношение только к предыдущей транзакции [5]. Вследствие этого и образуется цепочка блоков, она же блокчейн-система, в которой практически невозможна подмена предшествующих данных [5]. Именно благодаря этой особенности данная технология представляет интерес для многих исследователей.

Наиболее наглядной демонстрацией того, как функционирует блокчейн, является криптовалюта биткоин. За закрытие блока пользователь получает определенное количество биткоинов, которое уменьшается с каждым последующим закрытым блоком, но одновременно увеличивается комиссия за транзакцию. Конечно, люди не занимаются закрытием блоков вручную – для этого они используют компьютерные мощности, направляя их на решение сложных математических задач.

Однако блокчейн-технологии имеют потенциал не только в сфере финансов, но могут быть применены и в других сферах человеческой жизнедеятельности. Так, одним из авторов данной работы была выдвинута гипотеза, что технология блокчейн может быть применима для повышения надёжности мультиверсионного программного обеспечения (МВПО).

Мультиверсионное программное обеспечение предполагает независимую генерацию ряда функционально эквивалентных версий в соответствии с единой спецификацией программного обеспечения. При разработке мультиверсионного программного обеспечения в версиях реализуются различные методы и алгоритмы решения идентичных задач. Данный подход гарантирует, что ошибки одной из версий программного модуля не приведут к нарушению процесса работы всего программного обеспечения, что крайне важно для областей, для которых характерны повышенные требования по надежности. Таким образом, мультиверсионное исполнение программного обеспечения позволяет компенсировать и замаскировать сбои или отказы отдельных версий программных модулей и тем самым обеспечить отказоустойчивость и гарантировать выполнение целевых функций мультиверсионного программного обеспечения систем реального времени.

Вопрос повышения надёжности мультиверсионного программного обеспечения является актуальным, ввиду того, что данный класс программного обеспечения используется во многих сферах человеческой деятельности, где надёжность ПО критически важна, таких, например, как атомная энергетика, финансовый сектор, космические исследования, химическое производство и т.п.

Исследовательская гипотеза

Описание гипотезы

Концепция мультверсионного программного обеспечения подразумевает параллельное или последовательное выполнение набора функционально эквивалентных диверсифицированных программных компонентов (мультиверсий) одного модуля в единой среде исполнения [6]. Результаты вычислений программных компонент оцениваются, определяется их корректность, например, методами голосования. Результаты, признанные корректными, принимаются как результат работы всего модуля, к которому относились исполняемые мультиверсии. Результаты голосования будут тем точнее, чем больше версий реализуют данный модуль [7].

В настоящее время предложено множество алгоритмов голосования, которые различаются схемами работы и требованиями к исходным данным. Задача оценки достоверности голосования при использовании конкретного набора алгоритмов и задача выбора алгоритмов для имеющегося набора данных являются крайне актуальными. Можно отметить, что алгоритмы голосования, основанные на сравнении выходных данных версий, являются эффективными и интуитивно понятными. Однако применение таких алгоритмов требует разбиения множества выходов на подмножества, в которых элементы эквивалентны друг другу. Подобное разбиение в ряде случаев затруднено из-за проблемы несовместности разбиений. Очевидно, что результат работы алгоритма голосования определяет результат работы мультиверсионного программного обеспечения. Таким образом, решение проблемы обеспечения достоверности при голосовании в мультиверсионном программном обеспечении крайне важно.

Так как мультиверсии МВПО диверсифицированы по меньшей мере на одном из четырёх уровней [8], т.е. имеют значительные различия в своей реализации, потенциальные сбои в них независимы между собой. Поэтому даже если одна из версий выдаст ошибку, остальные – вернут корректный результат, что гарантирует безотказную работу всей системы [8].

Однако при использовании МВПО стоит также задача выявления и устранения возникающих во время работы мультиверсий сбоев. Именно для этого, а также для нахождения зависимостей между сбоями различных версий и модулей и предлагается использовать технологию блокчейн, которая будет применяться в качестве средства логирования (протоколирования работы).

В контексте данной статьи, логирование – это запись системной информа-

ции о работе мультиверсионной программной системы в файлы или цепочки блоков (логи). Данный процесс, протоколирующий все происходящие операции, позволяет разработчикам и администраторам получать наглядное представление о том, на каком этапе выполнения программы возник сбой, и, как следствие, предпринимать наиболее подходящие меры по его устранению [9].

Файловое логирование (протоколирование) или запись логов в базу данных не всегда являются безопасными, так как эти данные могут быть случайно или намеренно изменены или удалены кем-либо, в том числе злоумышленником. Кроме того, операции с файлами сами по себе имеют высокие риски потери или искажения данных [10]. Блокчейн же позволяет осуществлять логирование транзакций практически без возможности фальсификации или утери данных, благодаря чему увеличивается количество логов, вследствие чего повышается вероятность нахождения скрытой ошибки.

Разумеется, существует опасность того, что злоумышленники подключат к информационной сети предприятия своё оборудование, мощность которого хотя бы на 1% превышает мощности всего предприятия, чтобы иметь возможность исключения или искажения нежелательных для них логов [5]. Однако в рамках данной статьи этой возможностью было решено пренебречь, поскольку она относится хотя и к смежной, но всё же к отличной от представленной в работе тематике информационной безопасности. Причём вероятность корректного и полного логирования всех транзакций в системе в контексте данной работы априори принимается равной единице.

Теоретическое обоснование гипотезы

Поскольку ошибки в программном обеспечении могут быть как явными, так и скрытыми, к тому же могут проявляться на различных стадиях жизненного цикла программного обеспечения [11], необходимо определить, какие именно виды логов могут быть полезны для выявления и последующего устранения ошибок. Стоит отметить, что в контексте работы мультиверсионного программного обеспечения наибольший интерес представляют скрытые ошибки, которые могут проявляться лишь во время исполнения программы. Причём их проявление также может быть неявным. То есть конкретные сообщения об ошибках могут не выдаваться, однако некоторые логи могут свидетельствовать, например, о корреляции в работе мультиверсий и даже модулей.

Несмотря на то, что в каждом конкретном случае эксперт может по-разному классифицировать лог-сообщения, в зависимости от специфики решаемой задачи, в рамках данной работы было принято решение классифицировать логи следующим образом:

1) стандартные сообщения об ошибках, предусмотренные разработчиками мультиверсий – H_1;

2) сообщения о непредвиденных сбоях системы – H_2 (генерируются средой исполнения МВПО);

3) сообщения, неявно свидетельствующие о совместном использовании разделяемых аппаратных ресурсов несколькими версиями – H_3;

4) сообщения, неявно свидетельствующие о совместном использовании разделяемых аппаратных ресурсов несколькими модулями – H_4;

5) сообщения, неявно свидетельствующие об отсутствии корректного обмена данными между какими-либо версиями и средой исполнения МВПО – H_5;

6) все остальные сообщения – H_6.

Важно отметить, что сумма вероятностей нахождения ошибки каждого из перечисленных выше видов равна единице:

$$\sum_{i=1}^{m} P(H_i) = 1,$$

где m – количество классов сообщений (в нашем случае $m = 6$).

Положим, что A – это событие нахождения какой-либо скрытой ошибки в логах. Вероятность возникновения данного события $P(A)$ в данной работе было принято считать показателем, определяющим возможность повышения надёжности мультиверсионной программной системы, так как именно от возможности выявления ошибок на стадиях тестирования и эксплуатации ПО напрямую зависит возможность их устранения, что, в свою очередь, явным образом повышает надёжность мультиверсионного ПО.

Исходя из приведённых выше данных, становится возможным рассчитать вероятность обнаружения ошибки в работе программы по формуле полной вероятности:

$$P(A) = \sum_{i=1}^{m} P(A \mid H_i) P(H_i) = \sum_{i=1}^{m} P(AH_i),$$

где событие A может произойти при выполнении одного из событий H_i, т.е. при обнаружении одного из классифицированных сообщений по результатам анализа логов.

Вероятность появления ошибок для каждого класса сообщений может быть рассчитана по формуле Бернулли:

$$P_n(k) = C_n^k p^k (1-p)^{n-k},$$

где n – количество сообщений в лог-файле или цепочке логов, p – вероятность появ-

ления сообщения об ошибке в логах. Поскольку система является модульной, возможность параллельной работы версий и модулей не может быть исключена, из чего следует, что события записи логов считаются независимыми.

Результаты исследования и их обсуждение

Проведем исследование эффективности применения технологии блокчейн для задачи выявления скрытых ошибок. При работе мультиверсионного программного обеспечения в качестве критерия эффективности будем использовать значение вероятности нахождения скрытой ошибки посредством анализа логов. Скрытые ошибки в работе ПО можно диагностировать по ряду сообщений, заносимых в логах. Анализ логов результатов работы ряда мультиверсионных программных продуктов позволил выделить следующий ряд независимых сообщений, свидетельствующих об отклонениях в их работе:

1) предоставление итогового корректного результата работы одной из мультиверсий без вывода промежуточных данных;

2) превышение допустимого лимита времени работы нескольких мультиверсий;

3) увеличение объёма потребляемой модулем оперативной памяти;

4) невозможность среды исполнения МВПО обратиться к памяти, содержащей результат работы модуля;

5) вычислительные ошибки мультиверсий на некоторых наборах обрабатываемых данных.

Распространенность возникновения этих отклонений в работе программных средств задается следующими вероятностями:

$$P(H_1) = 0{,}00004;\ P(H_2) = 0{,}00002;$$
$$P(H_3) = 0{,}00003;\ P(H_4) = 0{,}000005;$$
$$P(H_5) = 0{,}00008.$$

Рассчитаем вероятность обнаружения скрытой ошибки $P(A)$. Анализ логов позволяет выявить одно из возможных приведенных сообщений об аномальном поведении в работе программного продукта. Значения вероятностей приведены в таблице.

Как видно из результатов, приведённых в таблице, цепочки логов, содержащие большее количество сообщений, позволяют с наибольшей вероятностью выявить скрытую ошибку и устранить её, что особенно актуально для стадии раннего тестирования.

Заключение

Основной целью применения мультиверсионной технологии является повышение надежности программного обеспечения и гарантированная его устойчивость к ошибкам. Избыточность и диверсификация версий критичных по надежности модулей программного обеспечения позволяет нивелировать последствия отказов отдельных компонент даже в случае возникновения ошибок, не выявленных на фазе тестирования программного обеспечения.

Применение технологии блокчейн может позволить выявлять не только логические ошибки, не выявленные на фазе тестирования, но и межверсионные, а также межмодульные зависимости на этапах тестирования и эксплуатации мультиверсионного программного обеспечения. Возможность точного выявления программного кода, содержащего ошибку, позволяет не только исправить возникшую неполадку, но и исключить дальнейший сбой работы программной компоненты при реализации всего набора мультиверсий. Чем больше мультиверсий выдают корректный результат, тем выше надежность мультиверсионного программного обеспечения.

Тем не менее логирование информации об исполнении мультиверсий программного обеспечения, наряду с увеличением надежности, также значительно увеличивает временные затраты на его разработку и доработку. За свою устойчивость технология блокчейн расплачивается низкой скоростью транзакций, что в свою очередь может производить негативный эффект [12]. Например, ощутимой эта проблема может оказаться при сбое в работе мультиверсионного программного обеспечения в какой-либо сфере человеческой деятельности, требующей экстренного вмешательства программиста [13, 14]. Также стоит отметить примитивность хранения данных: поиск возможен только по первичному ключу, а сам объем базы данных ограничен [12].

Результаты эксперимента

Число сообщений (n)	$P(AH_1)$	$P(AH_2)$	$P(AH_3)$	$P(AH_4)$	$P(AH_5)$	$P(A)$
100	0,0039842	0,0019960	0,0074442	4,9975e-004	0,0079369	0,017
250	0,0099	0,0049752	0,0074442	0,0012484	0,019606	0,06
500	0,019605	0,0099007	0,014777	0,0024938	0,038435	0,085
1000	0,038433	0,019604	0,029114	0,0049751	0,073855	0,17
10000	0,16375	0,090485	0,12911	0,024383	0,26815	0,68

Тем не менее, несмотря на данные недостатки, применение технологии блокчейн гарантированно обеспечивает повышение уровня надёжности мультиверсионного программного обеспечения вследствие более детального учета и анализа системных логов.

Список литературы

1. Андреев Е.В. Исследование возможности применения технологии «блокчейн» для защиты банковских транзакций // REDS: Телекоммуникационные устройства и системы. – 2017. – Т. 7, № 4. – С. 465–568.

2. Свон М. Блокчейн. Схема новой экономики / М. Свон. – М.: Olympus Business, 2017. – 240 с.

3. Генкин А.С., Михеев А.А. Блокчейн в интернете вещей // Страховое дело. – 2017. – № 10 (295). – С. 3-11.

4. Ярков В.В. Блокчейн и нотариат: опыт первой оценки // Нотариальный вестникъ. – 2017. – № 8. – С. 36–41.

5. Мухамадеева Р.И. Что такое блокчейн и как это работает? // Проблемы аграрной экономики в условиях импортозамещения: материалы междунар. науч.-практ. конф. – 2017. – С. 66–70.

6. Gruzenkin D.V., Chernigovskiy A.S., Tsarev R.Y. N-version Software Module Requirements to Grant the Software Execution Fault-Tolerance // Advances in Intelligent Systems and Computing. – 2017. – vol. 661. – С. 293–303.

7. Грузенкин Д.В., Новиков О.С., Суханова А.В. Мультиверсонное ПО и блоки восстановления – два способа защиты от ошибок // Новая наука: От идеи к результату. – 2016. – № 11–2. – С. 72–75.

8. Грузенкин Д.В., Якимов И.А., Кузнецов А.С., Царев Р.Ю. Определение метрики диверсифицированности мультиверсионного программного обеспечения на уровне алгоритмов // Фундаментальные исследования. – 2017. – № 6. – С. 36–40.

9. Шабанец Я.Р. Мультиплатформенный сервис логирования / Я.Р. Шабанец // Компьютерные системы и сети: материалы 49-й научной конференции аспирантов, магистрантов и студентов. (Минск, 6–10 мая 2013 года). – Минск: БГУИР, 2013. – С. 130–131.

10. Kernighan B.W., Pike R. The Unix programming environment. – Englewood Cliffs, NJ: Prentice-Hall, 1984. – Т. 270.

11. Вахрушева М.Ю., Евдокимов И.В. Показатели качества и надежности программного обеспечения // Труды Братского государственного университета. – 2012. – Т. 1. – С. 155–158.

12. Равал С. Децентрализованные приложения. Технология Blockchain в действии. – СПб.: Питер, 2017. – 240 с.

13. Царев Р.Ю., Штарик А.В., Штарик Е.Н., Завьялова О.И. Оценка транзакционной надежности современных систем управления и обработки информации // Приборы и системы. Управление, контроль, диагностика. – 2012. – № 6. – С. 29–32.

14. Евдокимов И.В. Проблема и показатели качества программного обеспечения // Труды Братского государственного университета. – 2009. – Т. 1. – С. 121–124.

15. Евдокимов И.В., Вахрушева М.Ю. Разработка программного обеспечения методик расчета показателей качества и надежности информационных систем обеспечения // Труды Братского государственного университета. – 2014. – Т. 1, № 1. – С. 192–196.