Код специальности ВАК 2.3.6
УДК 004

Бексаев Николай Сергеевич
Аспирант кафедры
«Информатика и информационная
безопасность»

Петербургский государственный
университет путей сообщения
Императора Александра I

Email: n.beksaev@yandex.ru
ORCID: 0000-0001-9660-3006
SPIN-код 9227-3569

ШАРДИНГ В БЛОКЧЕЙНЕ, ИЛИ РАЗДЕЛЯЙ И ВЛАСТВУЙ

Аннотация. Данная статья посвящена разработке способов решения одной из главных проблем всех современных блокчейн-платформ – проблемы масштабирования. Авторы рассматривают для этой цели использование сегментации сети блокчейн путем ее разделения на отдельные сегменты – так называемые шарды. Каждый шард, по сути, представляет собой отдельный блокчейн со своим собственным состоянием. Шардинг, несомненно, является одним из самых сложных, но и самым перспективным решением для обеспечения линейного роста производительности современных блокчейн-платформ.

Ключевые слова: шардинг, квадратичное шардирование, между-шардовые транзакции, оптимизация шардинга, центральный блокчейн

Abstract. This article explores solutions to one of the main challenges facing all modern blockchain platforms: scalability. The authors explore the use of blockchain network segmentation for this purpose, dividing it into separate segments called shards. Each shard is essentially a separate blockchain with its own state. Sharding is undoubtedly one of the most complex, yet most promising, solutions for ensuring linear performance growth in modern blockchain platforms.

Keywords: sharding, quadratic sharding, inter-shard transactions, sharding optimization, central blockchain

Введение

Хорошо известная трилемма блокчейна гласит, что в любой момент времени могут быть удовлетворены только два из трех основных свойств блокчейна: масштабируемость, безопасность и децентрализация. Авторы данной статьи в своих публикациях уже приводили пути решения этой трилеммы, которую авторы назвали трилеммой SSD блокчейна [7]. В данном исследовании авторы сосредоточили свое внимание на ключевой проблеме трилеммы SSD блокчейна – проблеме масштабируемости. Остановимся на этом подробнее.

Общеизвестным фактом является то, что одна из самых распространенных блокчейн-платформ, Ethereum, обрабатывает не более 20 транзакций в секунду [1]. Цена одной транзакции колеблется в широком диапазоне от 5 до 70 долларов США [2], а время ее подтверждения составляет от 16 секунд до 5 минут [3]. Несмотря на то, что блок в Ethereum публикуется в среднем раз в 22 секунды [4], время валидации одной транзакции до ее попадания в блок может составлять до 44 минут [5]. Таким образом, довольно

низкая пропускная способность, высокие затраты пользователей и длительное время валидации транзакций являются очень серьезным препятствием для запуска любых высокопроизводительных сервисов на Ethereum. Основная причина низкой производительности Ethereum заключается в том, что каждый узел в Ethereum должен проверять каждую транзакцию.

За пять лет с момента создания Ethereum было предложено несколько решений этой проблемы. Эти решения можно разделить на две группы:

- делегирование выполнения всех транзакций одному узлу или выделенной группе узлов с очень высокой производительностью;
- предоставление каждому узлу права обрабатывать только подмножество всех транзакций.

По словам разработчиков, использующих первый подход [6], удалось достичь 1200 транзакций в секунду, что в 60 раз больше, чем у Ethereum. Однако выполнение большого количества транзакций в конечном итоге всегда ограничено производительностью одного, пусть и очень мощного компьютера, что требует очень высокой надежности функционирования.

Второй подход называется «шардинг». Шардинг является одним из самых сложных решений в блокчейне; для его реализации предстоит провести еще много исследований и тестов. В частности, как показано на Рисунке 1, разработчики шардинга должны также создать механизм межшардовой коммуникации, который позволит смарт-контрактам на одном шарде взаимодействовать с другими смарт-контрактами на других шардах.

Терминология

Простейший шардинг Простейший шардинг поддерживает не один блокчейн, а несколько, и каждый из них называется «шардом». Каждый шард состоит из независимого подмножества узлов, участвующих в проверке транзакций и создании блоков. Далее мы будем называть такие узлы

валидаторами. Каждый шард обслуживает подмножество контрактов и счетов. Пока будем считать, что все транзакции всегда взаимодействуют только с контрактами и счетами в пределах одного шарда. Этого упрощения пока будет вполне достаточно для дальнейшего рассмотрения некоторых характерных проблем и особенностей шардинга.

Назначение валидаторов и центральный блокчейн Первая проблема заключается в том, что у каждого шарда есть свои валидаторы. Поэтому, если блокчейн содержит, например, 20 шардов, то каждый из шардов будет в 20 раз менее надежен, чем один блокчейн в целом. Представьте, что блокчейн содержит X валидаторов. Тогда система с 20 шардами равномерно распределит X валидаторов между 20 шардами. В этом случае в каждом шарде будет только X/20 валидаторов, и чтобы получить контроль над шардом, нужно получить контроль над 2,55% (51% / 20) валидаторов.

Возникает вопрос: как валидаторы назначаются шардам? Контроль над 2,55% валидаторов может стать проблемой только в том случае, если все они находятся в одном шарде. Если валидаторы не могут выбирать, в какой шард они назначены, получение контроля над 51% валидаторов до их назначения в шарды не позволит им получить контроль над каким-либо шардом. Ответ довольно прост – нужно использовать какой-то источник случайных чисел для назначения валидаторов в шарды. Для получения случайных чисел и назначения валидаторов вычислений в современных шардированных блокчейнах используется специально выделенный блокчейн, который существует исключительно для этих целей. В данной статье мы будем называть такой блокчейн «центральным блокчейном».

Квадратичное шардирование Использование центрального блокчейна тесно связано с концепцией «квадратичного шардирования». Шардинг часто ассоциируется с бесконечной масштабируемостью за счет увеличения числа

узлов. К сожалению, системы с центральным блокчейном имеют ограничение на количество шардов и не обладают бесконечной масштабируемостью, так как сам центральный блокчейн не шардирован, и поэтому его пропускная способность ограничена. Соответственно, количество шардов, поддерживаемых центральным блокчейном, ограничено.

Представьте, что мощность узла увеличилась в k раз. Каждый шард сможет обрабатывать в k раз больше транзакций, а центральный блокчейн сможет поддерживать в k раз больше шардов. Следовательно, пропускная способность всей системы вырастет в k2 раз. Отсюда и название «квадратичное шардирование». Однако, по мнению авторов, эту оценку следует считать слишком оптимистичной.

Шардинг состояния. Под состоянием шарда мы будем понимать всю информацию о счетах и контрактах. Узлы в блокчейне выполняют следующие три задачи:

- выполнение транзакций;
- пересылка транзакций и блоков другим узлам;
- хранение состояния и истории блокчейна.

Каждая из этих трех задач связана с постоянно растущей нагрузкой на узлы по следующим причинам:

- выполнение большего количества транзакций требует большей вычислительной мощности;
- пересылка большего количества транзакций требует большей пропускной способности сети;
- необходимость сохранения большего количества состояний и истории требует большего дискового пространства.

Тем не менее, отметим, что дисковое пространство не является самой большой проблемой. Сегодня состояние Ethereum занимает около 435 ГБ, что можно легко сохранить на любом современном компьютере.

Шардированные и нешардированные архитектуры

В нешардированной архитектуре каждый полный узел в сети блокчейн обрабатывает и проверяет каждую транзакцию. Типичными примерами нешардированных сетей являются Bitcoin и Ethereum. С другой стороны, в шардированной сети только достаточно большое подмножество сети будет обрабатывать и проверять данную транзакцию. Каждое из этих подмножеств узлов называется шардом. Преимущество шардинга заключается в том, что если исходную сеть можно разделить на несколько меньших подмножеств узлов, то каждое подмножество может обрабатывать неконфликтующие транзакции параллельно. Возможность параллельной обработки транзакций обеспечивает линейную масштабируемость.

Шардинг в отсутствие смарт-контрактов При отсутствии смарт-контрактов сеть может принимать только транзакции, подобные платежам, в которых, например, пользователь Алиса платит 5 токенов Бобу. Для простоты предположим, что в сети есть только два шарда, обозначенные S1 и S2. Итак, вопрос в том, должна ли транзакция от Алисы обрабатываться в S1 или S2? Очевидно, что транзакция не может быть назначена обоим шардам, иначе мы потеряем все преимущества шардинга.

- **Простое решение.** Простое решение — вычислить хэш транзакции и проверить последний бит хэша. Если последний бит хэша равен 0, то транзакция будет обработана в S1, в противном случае, если он равен 1, то транзакция будет обработана в S2. Эта стратегия назначения транзакций очень проста и может гарантировать, что нагрузка на обработку транзакций в S1 будет примерно такой же, как и в S2.

Однако у этого подхода есть проблема. Что, если у Алисы на счету было всего 5 токенов, и она создала дополнительную транзакцию, чтобы заплатить 5 токенов Чарли? С вероятностью 50% хэш второй транзакции будет иметь последний бит, отличный от хэша первой транзакции. Следовательно, они могут быть обработаны в разных шардах. Если два шарда не

взаимодействуют напрямую или через посредника, то становится невозможным обнаружить, что Алиса пытается провести атаку двойной траты.

Один из простых способов предотвратить атаки двойной траты — убедиться, что транзакции, приводящие к конфликтным ситуациям, обрабатываются в одном и том же шарде, что устраняет необходимость взаимодействия с другими шардами. Это можно легко сделать, изменив стратегию назначения. Вместо определения назначения на основе хэша транзакции, вы можете назначать транзакции на основе последнего бита адреса отправителя. Как следствие, все транзакции от отправителя попадают в один и тот же шард, и, следовательно, двойные траты могут быть обнаружены непосредственно в шарде.

Шардинг при наличии смарт-контрактов В этом случае давайте назначим транзакции на основе адреса отправителя, чтобы проверить, все ли работает как ожидалось. Сразу предупредим: эта стратегия назначения не будет работать со смарт-контрактами. Для примера возьмем следующий простой контракт. Важно то, что контракт должен иметь состояние, которое можно изменить. Приведенный ниже контракт имеет переменную состояния x, инициализированную значением 0, и контракт предоставляет набор интерфейсов для изменения значения x.

- **Простое решение I.** Применим стратегию назначения транзакций на основе адреса отправителя для транзакции T1, которая поступает от Алисы, в которой она вызывает функцию set с вводом 1. Предположим также, что эта транзакция попадает в шард S1, и шард изменяет значение x на 1. Предположим, пользователь Боб одновременно с Алисой выпускает другую транзакцию T2, в которой он вызывает set с вводом 2. Опять же, с вероятностью 50%, T2 может быть обработана в S2. Таким образом, T1 обрабатывается в S1, а T2 обрабатывается в S2. В этом случае S1 установит x в 1, а S2 установит y в 2. Возникает несогласованное состояние контракта. Необходим механизм

межшардового взаимодействия, чтобы избавиться от «грязного» состояния контракта.

- **Простое решение II.** А что, если мы изменим стратегию так, чтобы все транзакции, отправленные в смарт-контракты, назначались на основе адреса получателя, а не адреса отправителя? Новая стратегия назначения будет работать до тех пор, пока возможно разделять платежи и выполнение смарт-контрактов. Также необходимо учитывать случай, когда контракт вызывает другие контракты в виде цепочки, и конфликты могут возникать не обязательно из-за первого контракта. Таким образом, любая стратегия назначения должна быть общей, чтобы обрабатывать все контракты, не требуя слишком большого взаимодействия между шардами.

Комплексное решение. Попробуем классифицировать транзакции, чтобы определить отдельные стратегии их назначения для каждой категории. Транзакции, поступающие в сеть, можно разделить на следующие категории в зависимости от типа задействованных счетов:

- **Категория I [U1 → U2]:** Пользовательский счет, отправляющий часть токенов на другой пользовательский счет.
- **Категория II [U → C]:** Пользователь вызывает смарт-контракт, который не вызывает никаких других смарт-контрактов, и контракт не переводит средства другому пользователю.
- **Категория III [U1 → C1 → ... → Cn [→ U2]]:** Любая другая транзакция. Эта категория включает транзакции, исходящие от пользователя, которые могут вызывать цепочку контрактов и потенциально завершаться с использованием пользовательского счета.
- **Не очень оптимизированная версия** Далее в статье авторы без ограничения общности предполагают, что в любой момент времени в сети существует k = 2n шардов, где n – натуральное число. В любой момент времени каждый шард будет обрабатывать только транзакции категорий I и II, в то время как специальный шард будет выполнять

только транзакции категории III строго после того, как транзакции категорий I и II будут подтверждены остальными шардами. Стратегия назначения следующая: для сети с k = 2n шардами каждый шард будет обрабатывать только транзакции категорий I и II, в которых адреса отправителя и получателя имеют одинаковые последние n битов. Любая другая транзакция (включая категорию III) будет обрабатываться специальным шардом после того, как другие шарды завершат обработку предназначенных для них транзакций. Эта не очень оптимизированная версия создает довольно большую нагрузку на специальный шард.

- **Оптимизированная версия** Мы применяем ту же стратегию назначения, что и раньше, за исключением того, что мы разрешаем назначать платежные транзакции категории I по адресу отправителя. Итак, в общем, стратегия назначения будет следующей: для сети с k = 2n шардами каждый шард будет обрабатывать транзакции категории II, в которых адреса отправителя и получателя имеют одинаковые последние n битов. Каждый шард также будет обрабатывать транзакции категории I с одинаковыми последними n битами адреса отправителя. Любая другая транзакция (включая категорию III) будет обрабатываться специальным шардом только после того, как шарды завершат обработку транзакций категорий I и II. Очевидно, что оптимизированная версия обеспечивает гораздо лучшую масштабируемость, чем неоптимизированная. Для обеих стратегий, чем больше шардов, тем больше транзакций потребуется обработать специальному шарду.

Транзакции между шардами

Если участник на каком-то шарде хочет перевести токены участнику на том же шарде, валидаторы этого шарда могут обработать эту транзакцию и применить ее к состоянию. Но, например, если у Алисы есть счет на шарде S1, и она хочет отправить токены Бобу со счетом на шарде S2, ни валидаторы шарда S1 (которые не смогут добавить токены Бобу), ни валидаторы шарда S2

(которые не смогут забрать токены Алисы) не могут завершить транзакцию и обновить состояние

Существуют две большие группы подходов к решению этой проблемы:

- **Синхронный:** Для любой транзакции, затрагивающей несколько шардов, блоки в шардах, содержащие обновление состояния для этой транзакции, создаются одновременно, и валидаторы в этих шардах работают вместе для создания таких блоков.
- **Асинхронный:** Межшардовая транзакция выполняется асинхронно в затрагиваемых ею шардах. Проблема этого подхода в том, что если блоки создаются независимо, то есть вероятность, что один из блоков, содержащих обновление состояния для транзакции, не окажется в канонической цепочке своего шарда, и, таким образом, транзакция будет выполнена лишь частично. Этот сценарий является одной из ключевых проблем в шардинге.

О безопасности шардинга

С ростом популярности протоколов консенсуса, основанных на алгоритме PoW, возрастает вероятность угроз для шардированного блокчейна. Это связано с тем, что PoW зависит от вычислительной мощности, а при использовании шардинга она уменьшается для каждого шарда, что позволяет злоумышленникам получить над ним контроль. Вероятность атаки на сеть возрастает, поскольку вычислительная мощность, необходимая для установления контроля над определенным шардом, теперь намного ниже, чем мощность, необходимая для контроля всей сети. В случае блокчейна в целом, основанного на алгоритме консенсуса PoW, такая атака практически невозможна из-за огромной требуемой вычислительной производительности.

Если в блокчейне с шардингом используется алгоритм консенсуса PoW, атака 51% вполне возможна и представляет реальную угрозу безопасности сети. Злоумышленники могут сосредоточить свои усилия только на одном шарде и полностью взять его под контроль. Затем, используя

протокол межшардового взаимодействия, они могут атаковать другие шарды сети блокчейн.

Для обеспечения безопасности, децентрализации и масштабируемости виртуальной машины Ethereum необходимо было перейти на алгоритм консенсуса Proof-of-Stake (PoS). Алгоритм PoS не позволяет хакерам концентрироваться на атаке одного шарда благодаря эффективному использованию случайной выборки. Система на основе алгоритма PoS устранит риск атаки 51%, который останется, если Ethereum продолжит работать на алгоритме PoW. В случае блокчейна с шардингом на основе алгоритма PoS потенциальные злоумышленники не смогут выбрать конкретный шард и заранее узнать, с какой частью сети они будут работать. Такой подход устранит риски, связанные с внедрением шардинга.

Заключение

В данной статье авторы осветили актуальные проблемы шардинга блокчейна. Первоначальный подход, основанный на протоколах консенсуса PoW, уступает место протоколам PoS в самых популярных блокчейнах. Эти протоколы предусматривают наличие определенного количества узлов-валидаторов в каждом из шардов. При выполнении межшардовых транзакций возникает ряд проблем, вызванных возможным захватом некоторых валидаторов злоумышленниками и построением альтернативной цепочки блоков. На сегодняшний день надежного и универсального решения этих проблем не существует, однако авторы в своем исследовании предложили некоторые из наиболее актуальных путей.

Внедрение шардинга все еще находится на ранней стадии, и многие вопросы и технические проблемы еще предстоит решить. В то же время шардинг является перспективным решением для улучшения масштабируемости сетей блокчейн, хотя и создает серьезные технические и реализационные проблемы, которые необходимо решить. Надежное решение проблем шардинга позволит создавать быстрые и доступные блокчейн-протоколы – важный компонент в инфраструктуре открытого Интернета.

Список литературы

[1] “The number of Ethereum transactions is only 20 per second. Scaling problem”, https://bytwork.com/articles/scaling, Accessed: February 07, 2023.

[2] “Ethereum Average Commission Chart”, https://bitinfocharts.com/ru/comparison/ethereum-transactionfees.htmlS3y, Accessed: February 07, 2023.

[3] “How long does it take to send Ethereum?”, https://foreck.info/blog/skolko-vremeni-nuzhno-chtoby-otpravit-ethereum, Accessed: February 07, 2023.

[4] “Ethereum Block Time chart”, https://bitinfocharts.com/ru/comparison/ethereum-confirmationtime.htmlS3y, Accessed: February 07, 2023.

[5] “The average transaction processing time in the Ethereum network has reached 44 minutes”, https://coinspot.io/technology/ethereum/srednee-vremya-obrabotki-tranzakcii-v-seti-efiriuma-dostiglo-44-minut, Accessed: February 07, 2023.

[6] “The mobile entertainment revolution begins on ThunderCore”. https://www.thundercore.com/, Accessed: February 07, 2023.

[7] Kustov V, and Beksaev N, “The Blockchain SSD Trilemma or Chasing Three Birds with One Stone”, Proc. Of Microwave and Radio Electronics Week 2023, Pardubice, Czech Republic, April 19-20, 2023.