Анализ надежности распределенной системы хранения файлов на основе блокчейн

О. Хаммуд[I], И.А. Тарханов[II,III]

[I] Национальный Исследовательский Технологический Университет (НИТУ) «МИСиС», г. Москва, Россия
[II] Федеральное государственное бюджетное учреждение высшего образования «Государственный академический университет гуманитарных наук», г. Москва, Россия
[III] Федеральное государственное учреждение «Федеральный исследовательский центр «Информатика и управление» Российской академии наук», г. Москва, Россия

Аннотация. В данной работе производится оценка надежности системы хранения файлов DecStore, описанной в серии статей [1–3]. Эта распределенная система использует блокчейн в качестве децентрализованного балансировщика нагрузки и распределяет файлы по узлам с помощью виртуальных дисков. При этом общее количество необходимого места для обеспечения надежности функционирования всей системы меньше системы полного резервирования данных на 25%. Однако рассчитать надежность такой системы известными методами нельзя, т.к. они не учитывают время автоматического восстановления данных на других узлах и максимально возможное количество восстановлений после потери одного узла. Предлагается математическая модель для расчета надежности такого класса систем. Приведен результат расчета с применением представленной модели и на основе полученных результатов делан вывод о преимуществах системы балансировки нагрузки на базе смарт-контрактов.

Ключевые слова: *блокчейн, p2p, распределенное файловое хранилище, надежность, балансировщик нагрузки.*

DOI: 10.14357/20790279230402 **EDN: YLCMKG**

Введение

С ростом популярности блокчейн-технологии и интереса к разработке приложений и систем на базе закрытых и открытых блокчейн-сетей (DApps) возникает вопрос об оценке целесообразности применения таких решений с учетом их известных преимуществ (гарантия неизменности данных, конфиденциальность), так и недостатков (перерасход ресурсов, сложность поддержки и развертывания). Один из важных аспектов этой проблемы является оценка надежности Dapps. В этой работе мы рассмотрим подробнее надежность распределенной системы обмена файлами, которая управляется через реализованные в блокчейн смарт-контракты. Такая система позволит организовать надежное хранение и передачу файлов между рядом организаций или филиалами крупного холдинга, которые не доверяют друг другу. Например, это может быть сеть страховых компаний, оценщиков и крупных ремонтных предприятий, либо электронная торговая площадка крупного холдинга, которая собирает предложения, ищет поставщиков и заключает с ними контракты.

Анализ существующих решений показал, что они обладают следующими недостатками:

- Наличие единой точки отказа в архитектуре. Например, для проектов HDFS[4] и Luster [5] – это балансировщик нагрузки или веб-сервер для подключения по API.
- Множественный перерасход ресурсов. Распределенные системы многократно копируют данные на всех узлах системы. Так работают классические открытые блокчейн-платформы, torrent-обменники. Для широкого внедрения нужны системы, которые тратят меньше дискового пространства, но при этом сохраняют высокую доступность.

- Неравномерность распределения данных. Важным моментом является равномерное распределение ресурсов по системе. В большинстве случаев отсутствует процесс распределения между серверами тех или иных данных, что приводит к появлению «узких мест».
- Ограничения по расширяемости. Важным преимуществом является возможность развертывания новых узлов на любых технических средствах с минимальным количеством затрат. Это позволяет быстро масштабировать системы горизонтально без потери свойств надежности и доступности.

Ранее нами была представлена архитектура системы [3] (далее DecStore), которая позволит избежать этих недостатков. Но внедрение такой системы возможно после понимания ее надежности в сравнении с другими подходами, которые решают похожие задачи – RAID 5 или RAID 6 [6], HDFS, Luster или традиционные системы с полным бекапированием файлов на другие сервера (зеркалированием).

Нужно отметить, что надежность распределенных систем хранения данных, построенных на базе блокчейн (on-chain), P2P или централизованных архитектур (off-chain), недостаточно изучена на данный момент. В [7] обсуждаются теоретические аспекты и не предлагается единой методики расчета таких систем. В [8] обсуждаются системы с разной моделью распределения данных, но не вычисляется их надежность целиком. Использовать известные методы для расчета надежности (например, [6,9]) нельзя, т.к. они не учитывают возможности автоматического масштабирования и распределения данных по разным узлам, а также максимально возможное количество восстановлений после потери узла.

Целью данной статьи является разработка математической модели и проведение расчета надежности системы распределенного хранения файлов на основе смарт-контрактов. Также проводится обсуждение результатов в сравнении с другими известными технологиями хранения файлов на множестве объединенных носителей. При этом они могут быть объединены в рамках одного физического сервера (например, RAID [6] или за счет технологии виртуализации), так и нескольких серверов, связанных сетью.

1. Архитектура системы распределенного хранения файлов на виртуальных дисках с Erasure coding на базе смарт-контрактов

Архитектура DecStore состоит из трех основных компонентов (рис. 1).

I. Балансировщик нагрузки обеспечивает необходимые функции управления и распределения данных. Он решает следующие задачи.

1. Управление узлами включает в себя управление существующими узлами, контроль процессов присоединения и выхода узлов и т. д.
2. Управление данными включает в себя управление размещением существующих файлов, определение места добавления новых файлов. Предлагаемая модель хранения использует виртуальные объекты для хранения данных, которые называются виртуальными кластерами (VC) и виртуальными дисками (VD), что будет объяснено в разделе, посвященном хранению данных. Балансировщик нагрузки отвечает за управление этими виртуальными объектами путем их распределения, управления расположением новых виртуальных дисков и т. д. Балансировщик нагрузки обрабатывает распространение и восстановление файлов, а также операции, связанные с узлами, с использованием новых предложенных алгоритмов, которые будут обсуждаться далее в этой статье.
3. Управление пользователями включает добавление и удаление пользователей, а также управление их привилегии.

В DecStore блокчейн используется для запуска балансировщика нагрузки в виде смарт-контракта. В этом случае пользователи могут подключаться к любому узлу блокчейн-сети, а все узлы синхронизируются благодаря алгоритму консенсуса блокчейн-сети.

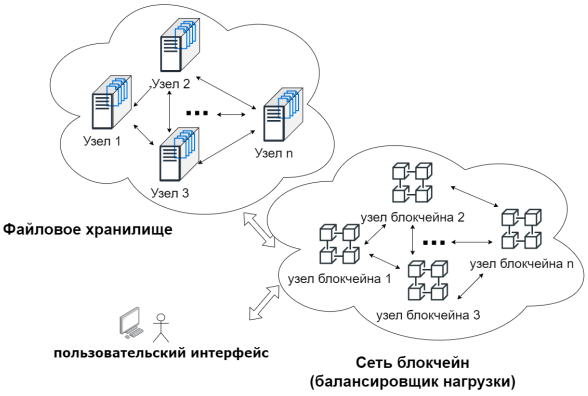

Рис. 1. Архитектура системы

II. Система хранения файлов состоит из узлов (сеть P2P, где все узлы имеют одинаковую роль). Файлы организованы в виртуальных кластерах (VCs). В свою очередь, каждая из которых состоит из трех виртуальных дисков (VD) фиксированного

размера. Это сделано для того, чтобы при масштабировании алгоритмы управления распределением файлов по виртуальным дискам и кластерам имели меньшую сложность, что важно, т.к. они реализованы на смарт-контрактах.

Когда файл загружается пользователем или внешней информационной системой на диск, то он делится на две части и вычисляется erasure coding – последовательность байт, по которой можно восстановить одну из частей этого файла при наличии другой [10]. Важно отметить, что размер erasure coding равен половине файла. Далее вызывается балансировщик нагрузки (метод смарт-контракта в блокчейн), который выбирает один VC (три VDs), где больше всего свободного места. Далее происходит запись двух частей файла и erasure coding на соответствующие виртуальные диски, если данный кластер еще не достиг максимального размера. На каждый VD запись новой части файла происходит в виде отдельного файла с соответствующим разрешением. Например, первая часть файла document123.01, вторая часть document123.02, а erasure coding - document123.ec. Это помогает легче выполнить поиск и удаление файлов если понадобится перераспределение файлов по VD в будущем.

В каждом кластере первый VD всегда используется для записи первых половин файлов, второй VD – для вторых половинок, а третий – для erasure coding. Например, на рис. 2,A в VD6-P1 записывается первая часть файла, в VD6-P2 – вторая часть, а в VD6-EC – вычисленный erasure coding. При потере одной из трех частей, из оставшихся можно восстановить все файлы виртуального диска (p1&p2 или p1&EC, или p2&EC). VD6-P1, VD6-P2 и VD6-EC составляют VC6.

Если не хватает места для добавления нового файла на виртуальном кластере, то балансировщик нагрузки принимает решение, где создать новый виртуальный кластер, используя алгоритм распределения данных в смарт-контракте.

В случае, если один из узлов выходит из строя (по факту ни один блокчейн-узел не может связаться с ней в течении какого то времени), балансировщик нагрузки выбирает, где можно восстановить потерянные VDs, используя алгоритм восстановления. На рис. 2,B показан случай отказа узла 4, а на рис. 2,C – узла 5 после восстановления узла 4. Восстановленные VD с узлов 4 и 5 выделены жирным.

Существует несколько условий, по которым определяется, на каком узле надо записать восстановленный VD – узел не должен содержать других

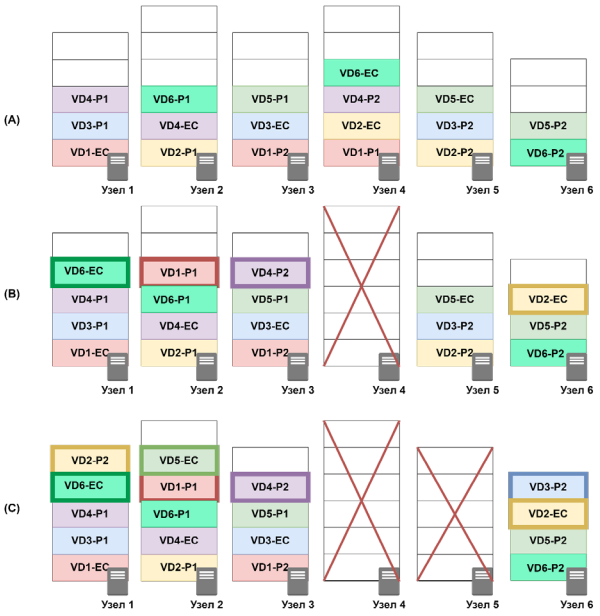

Рис. 2. Восстановление данных в случае отказа узлов

VD из той же VC и учитывается количество свободного места на каждом узле.

VD, которые используют для erasure coding, можно рассматривать как «холодные». Они помогают сбалансировать нагрузку на систему, так как новые виртуальные диски erasure coding считаются расположенными на узлах, имеющих наибольшую нагрузку. Эти виртуальные диски вызываются только тогда, когда требуется восстановить потерянные данные. Виртуальные диски, к которым чаще всего обращаются, можно рассматривать как «горячие». Они учитываются при создании или перераспределении виртуальных дисков, чтобы сбалансировать нагрузку на систему.

Процесс восстановления данных повторяется до тех пор, пока во всей системе не останется места для восстановления потерянных дисков или не останется всего три узла.

III. Интерфейс позволяет пользователю взаимодействовать с системой. Пользовательский интерфейс также содержит разделитель файлов, вместо того чтобы включать его в хранилище данных, чтобы уменьшить нагрузку на систему.

В примере, показаном на рис. 2, невозможно дальше восстановить узел, т.к. не хватает места. Здесь максимальное количество процессов восстановления равно 2.

Для поддержания балансировки нагрузки в системе были разработаны четыре алгоритма:

1) Алгоритм добавления новых узлов (A1).
2) Алгоритм загрузки файлов (A2).

3) Алгоритм восстановления узла (А3).
4) Алгоритм возобновления узла (А4).

Особенности работы данных алгоритмов описаны в [3] и реализованы с условиями ограничений работы в chaincode Hyperledger Fabric [11,12].

2. Расчет надежности

Перейдем к расчету надежности рассматриваемой системы. Балансировщик нагрузки может размещаться на любом, в том числе и внешнем узле в блокчейн-сети, или включен в саму систему хранения (располагаться на тех же серверах, что и виртуальные диски для файлов). Далее рассматривается такая конфигурация системы, где блокчейн и смарт-контракт размещены на каждом сервере, вместе с файлами. В такой конфигурации при выходе из строя одного из узлов блокчейна можно подключатся к любой другой, которые содержит всю цепочку данных. Отказ системы произойдет только при выходе из строя всех узлов блокчейн. Следовательно, надежность всей системы зависит от подсистемы хранения файлов. Рассмотрим ее подробнее.

Введем следующие определения:

- $\boldsymbol{N}$ – множество узлов в системе;
- $\boldsymbol{VD_S}$ – множество виртуальных дисков в системе;
- $\boldsymbol{VD`_S}$ – множество пустых слотов виртуальных дисков в системе;
- $\boldsymbol{VC}$ – множество виртуальных кластеров в системе;
- $\boldsymbol{VD_{n(i)}}$ – подмножество $\boldsymbol{VD}$, расположенных в узле i;
- $\boldsymbol{VD_{n(i,x)}}$ – подмножество $\boldsymbol{VD}$, расположенных в узле i, которые имеют состояние x;
- $\boldsymbol{VD`_{n(i)}}$ – подмножество $\boldsymbol{VD`}$, расположенных в узле i;
- $\boldsymbol{VD_{index(i,j)}}$ – элемент множества $\boldsymbol{VD}$, расположенный в узле i и имеющий порядок j;
- $state(d)$ – функция, возвращающая состояние виртуального диска d. Выход (вывод) представляет одно из состояний hot(горячим), cold(холодным) или normal (обычным);
- $\boldsymbol{VD_{S(x)}}$ – подмножество $\boldsymbol{VD}$, которое имеет состояние *x;*
- $vd_{vc,p}$– элемент множества $\boldsymbol{VD}$, представляющий vd, принадлежащий виртуальному кластеру vc и имеющий часть p. p может быть 1 для первой части, 2 – для второй части, 3 – для Erasure coding;
- $\boldsymbol{VC_{n(i)}}$ – подмножество $\boldsymbol{VC}$, которым принадлежат узлы i;
- $\boldsymbol{VC_i}$ – элемент множества $\boldsymbol{VC}$ с индексом i;
- $fsr(i)$ (Free space to request ratio) – функция, возвращающая отношение свободного места к запросу для узла i;
- vc_{vd} – элемент множества $\boldsymbol{VC}$, который содержит конкретный виртуальный диск d;
- $p(d)$ – функция, возвращающая номер части виртуального диска d;
- $mcvd(x)$ (Mean count of VDs) – функция, которая возвращает среднее количество VD в состоянии x на узел.

Для вычисления надежности системы в течении года (8760 часов) нужно вычислить вероятность отказа всей системы. Рассматриваемая надежность системы противоположна вероятности отказа. Итак, это 1 – вероятность отказа, вторую можно измерить с помощью AFR (Annualized Return Rate) [13,14]:

$$AFR = 1 - e^{\frac{8760}{MTTF_{system}}}. \quad (1)$$

Поскольку $8760/MTTF_{system} \ll 1$, AFR, T можно представить как:

$$AFR \approx \frac{8760}{MTTF_{system}}. \quad (2)$$

Таким образом, R можно рассчитать следующим образом:

$$R = 1 - \frac{8760}{MTTF_{system}}. \quad (3)$$

Для вычисления надежности нужно вычислить время наработки на отказ всей системы (MTTF – Mean time to fail), обозначим его как $MTTF_{system}$, а $MTTF_{node}$ – время наработки на отказ для конкретного узла. $MTTF_{system}$ и $MTTF_{node}$ измеряются в часах, а R не имеет единицы измерения.

Мы предполагаем, что система выходит из строя, когда часть хранящихся данных безвозвратно потеряна. Т.е. достаточно привести ее в состояние, когда будет потерян хотя бы один файл из VD. $MTTF_{system}$ можно рассчитать, воспользовавшись формулой из расчета надежности для RAID [6]:

$$MTTF_{system} = \frac{MTTF_{node}}{|\boldsymbol{N}|} * \frac{1}{P_{SFANF}}, \quad (4)$$

где $\boldsymbol{N}$ – множество узлов в системе; P_{SFANF} (SFANF: System fails after the node has failed) – вероятность, что система откажет после отказав ней одного узла.

Поскольку DecStore реализует динамическое восстановление на остальных узлах, P_{SFANF} рассчитывается иначе, чем для RAID, так как требуется учитывать возможное время восстановления данных на основе свободного места, оставшегося

в системе. Чтобы рассчитать надежность системы, предположим, что произвольный узел вышел из строя и нужно определить вероятность отказа всей системы после потери этого узла. Например, в системе с полным резервным копированием, в которой есть один сервер резервного копирования для каждого узла, система может выйти из строя при условии, что до восстановления первого потерянного узла на новом сервере произойдет потеря еще одного узла.

Рассмотрим подробнее от чего зависит P_{SFANF}. Например, вероятность потери других узлов в процессе восстановления P_{NFBR} (Node Failure Before the Repairment is finished):

$$P_{NFBR(i)} = \frac{MTTR_i}{MTTF_{node(i)}/|\boldsymbol{N}`|}, \qquad (5)$$

где $\boldsymbol{N}`$ – множество оставшихся узлов, которые участвуют в восстановлении потерянного узла.

Приведем пример для систем, где осуществляется полное резервное копирование, где для каждого узла существует два сервера резервного копирования. В это случае, $\boldsymbol{N}`$ = 2 для $i$=1 (если один узел выходит из строя). Если 2 узла выходят из строя ($i$=2), то можно восстановиться с первого узла, т.е. для $i$=2, $\boldsymbol{N}`$ = 1. Необходимо заметить, что в системах полного резервного копирования не важно общее количество узлов. Имеет значение только количество резервных для каждого узла.

Поскольку разные узлы могут иметь разные характеристики, расчеты $P_{NFBR(i)}$, $MTTR_i$, $MTTF_{node(i)}$ различаются в зависимости от узла i. Будем считать, что все узлы имеют одинаковые характеристики для простоты вычислений, поэтому вместо этого будем использовать $P_{NFBR}, MTTR, MTTF_{node}$

Практически, $MTTR \ll \frac{MTTF_{node}}{|N`|}$, так что время, необходимое для ремонта/замены диска, намного меньше, чем его срок службы.

Для $MTTR_{node}$ время восстановления узла в часах (mean time to repair a node) системы полного бекапирования вычисляется следующим образом:

$$MTTR_{node} = \frac{S_{node}}{Th * 3600 * R_{rate}} + Tr \qquad (6)$$

где S_{node} – размер хранилища определенного узла; Th – пропускная способность узла в Мб/сек.[1] Предположим, что $MTTR_{node}$ одинаково для всех узлов; R_{rate} (rebuild rate) показывает сколько нужно выделить системных ресурсов для восстановления утраченного диска; среднее значение 50%; Tr – время, необходимое для замены потерянного узла в часах. Оно равно 0 если используется запасной узел, который уже подключен к сети и сконфигурирован, как основной для любого узла (скопирован из резервного узла в случае систем полного резервирования). Так называемый «горячий» узел.

Так как система состоит из множества узлов можно предположить, что общий размер дисков всех узлов одинаковый (S) и можно вычислить обобщенный *MTTR* для всех узлов:

$$MTTR_{node} = \frac{S}{Th * 3600 * R_{rate}} + Tr\ . \qquad (7)$$

В предложенной модели, когда узел выходит из строя, ее данные перестраиваются на оставшихся узлах.

Время восстановления в предложенной системе:

$$MTTR_{node} = \frac{S}{|\boldsymbol{N}`| * Th * 3600 * R_{rate}} + Td, \qquad (8)$$

где Td – время распределения потерянных *VD* по оставшимся узлам в часах. В более общем случае это может быть время восстановления блока данных (dataUnit), файла, части файла фиксированного размера, или как в нашем случае, виртуального диска.

При этом сложность алгоритма распределения $O(|VD_{lostNode}|)$(количество vds в потерянном узле) ,чем меньше количество блоков данных, тем быстрее он работает:

$$Td = \sum_{dataUnit=1}^{|dataUnits|} Td_{dataUnit}\ , \qquad (9)$$

где $dataUnits$ – набор всех блоков данных.

Нужно отметить, что чем меньше размер блока данных мы используем (например, используем файл вместо виртуального диска), тем больше значение |*dataUnits*|. Следовательно, по формуле (8) увеличивается *Td*, а значит увеличивается *MTTR* и уменьшается *MTTF*.

Однако группировка данных в единый большой виртуальный диск не приводит к увеличению надежности, т.к. достаточно затратно иметь другой такой же узел, который будет иметь необходимое свободное место, что делает применение такого подхода бессмысленным по сравнению с системой полного резервного копирования.

При использовании виртуальных дисков, а не отдельных файлов, алгоритм восстановления работает намного быстрее, т.к. ему надо вычислить оставшиеся части только для потерянных

[1] Т.к. $MTTF_{node}$ измеряется в часах, то *Th* необходимо умножить на 3600.

VD, а не каждого из тысячи или миллиона файлов, которые были на вышедшем из строя узле. Размер *VD* нужно выбрать так, чтобы он был достаточным для уменьшения параметра |*dataUnits*| и достаточно малым для удобства размещения на оставшихся узлах.

Для расчета $MTTF_{node}$ предположим, что у каждого узла всего один жесткий диск, тогда мы в последствии сможем более корректно сравнить предложенный подход с другими известными системами хранения. В этом случае $MTTF_{node}$ определяется производителем конкретного диска. Например, для дисков, которые используются для файловых хранилищ корпоративного уровня $MTTF$ обычно равен 2000000 часов [15].

Предложенная система имеет динамическую модель восстановления, где при выходе узла из строя, все данные на нем могут быть восстановлены на оставшихся узлах *MPRC* раз. (Maximum Possible Recoveries Count) определяется как счетчик максимально возможного количества восстановлений после потери. Процесс восстановления возможен, когда два из следующих условий выполняется:

$$\left|\bigcup_{i=1}^{|N|}\left(\boldsymbol{VD`_{n(i)}}\right)\backslash\{\boldsymbol{VD`_{n(j)}}\}\right| \geq \left|\boldsymbol{VD_{n(j)}}\right| \forall\, 1 \leq j \leq \boldsymbol{N}\ ; \quad (10)$$

$$|\boldsymbol{N}| - 1 \geq 3\ . \quad (11)$$

Следовательно, можно восстановить данные, пока будет достаточно свободного места во всей системе для восстановления любого вышедшего из строя узла и пока будет не меньше трех узлов в системе.

Важно отметить, что условие (10) включает только свободное место, которое может содержать новые виртуальные диски. Не все свободное место на узлах можно использовать. Представим, что требуется восстановить потерянный *VD* размером 1 Гб. При этом в системе осталось 4 узла на каждом из которых по 512 Мб. Система не сможет восстановить *VD*, не смотря на то, что сумма свободного места на оставшихся узлах равна 2 Гб. Предполагая, что $\boldsymbol{VD`_S}$ – это набор свободных слотов VD в системе, условие можно переписать следующим образом:

$$\left|\boldsymbol{VD`_S} \setminus \boldsymbol{VD`_{n(j)}}\right| \geq \left|\boldsymbol{VD_{n(j)}}\right| \forall\, j \in \boldsymbol{N}\ . \quad (12)$$

Следовательно, $MPRC$ может быть определено следующим образом:

$$MPRC = max\{n = 1..|\boldsymbol{N}| - 2 : F(n) \geq 0\} + 1; \quad (13)$$

$$F(n) = F(n-1) - \frac{|\boldsymbol{VD_S}|}{|\boldsymbol{N}| - n + 1} - \frac{F(n-1)}{|\boldsymbol{N}| - n + 1}; \quad (14)$$

$$F(0) = |\boldsymbol{VD_S}|\ , \quad (15)$$

где *F(n)* –рекурсивная функция, которая вычисляет свободное место в системе на основе количества вышедших из строя *n* узлов.

MPRC равно максимальному количеству процессов восстановления + 1, потому что даже если больше нет свободного места для повторного восстановления данных, будет возможность восстановить данные на стороне пользователя (данные все еще могут быть доступны).

В системе с полным резервным копированием, если существует *B* резервных серверов для каждого узла, произойдет отказ всей системы, если все рабочие и резервные сервера выйдут из строя до копирования данных на новые сервера. Эту модель можно определить как:

$$P_{SFANF} = \prod_{i=1}^{B} \frac{MTTR_i}{\frac{MTTF_{node}}{B+1-i}} = B! * \prod_{i=1}^{B} \frac{MTTR_i}{MTTF_{node}} \quad (16)$$

Тогда $MTTF_{system}$ вычисляется в системах полного резервного копирования как:

$$MTTF_{system} = \frac{MTTF_{node} * B!}{|\boldsymbol{N}|} * \prod_{i=1}^{B} \frac{MTTF_{node}}{MTTR_i}. \quad (17)$$

В DecStore вся система выйдет из строя в одном из следующих случаев:

- если 2 узла выйдут из строя до того времени пока не кончится время, необходимое для восстановления одной из вышедших из строя узлов на оставшиеся в системе узлы;
- если 3 узла выйдут из строя до того пока не пройдет необходимое время для восстановления двух из них на оставшиеся в системе узлы;
- произойдет выход из строя такого количества узлов, которое равно *MPRC*.

Важно отметить, что вероятность отказа системы в случае *MPRC* = 3, вычисляется следующим образом:

$$\begin{aligned} P_{SFANF} &= \frac{MTTR_1}{MTTF/(|\boldsymbol{N}|-1)} \\ &+ \left[\frac{MTTR_1 + MTTR_2}{MTTF/(|\boldsymbol{N}|-1)} * \frac{MTTR_1 + MTTR_2}{MTTF/(|\boldsymbol{N}|-2)} - \frac{MTTR_1}{MTTF/(|\boldsymbol{N}|-1)}\right] \\ &+ \left[\frac{MTTR_1 + MTTR_2 + MTTR_3}{MTTF/(|\boldsymbol{N}|-1)} * \frac{MTTR_1 + MTTR_2 + MTTR_3}{MTTF/(|\boldsymbol{N}|-2)}\right. \\ &\left. * \frac{MTTR_1 + MTTR_2 + MTTR_3}{MTTF/(|\boldsymbol{N}|-3)} - \frac{MTTR_1 + MTTR_2}{MTTF/(|\boldsymbol{N}|-1)} * \right. \\ &\left. * \frac{MTTR_1 + MTTR_2}{MTTF/(|\boldsymbol{N}|-2)}\right] \end{aligned} \quad (18)$$

Или

$$P_{SFANF} = \left[\frac{MTTR_1 + MTTR_2 + MTTR_3}{\frac{MTTF_{node}}{(|\boldsymbol{N}| - 1)}} * \right.$$

$$\left. * \frac{MTTR_1 + MTTR_2 + MTTR_3}{\frac{MTTF_{node}}{(|\boldsymbol{N}| - 2)}} * \frac{MTTR_1 + MTTR_2 + MTTR_3}{\frac{MTTF_{node}}{(|\boldsymbol{N}| - 3)}}\right] \quad (19)$$

Следующая формула представляет обобщенный вид вычисления вероятности отказа всей системы:

$$P_{SFANF} = \left(\frac{MTTR * MPRC}{MTTF_{node}}\right)^{MPRC} * \prod_{i=1}^{MPRC} (|\boldsymbol{N}| - i) \quad (20)$$

Следовательно, $MTTF_{system}$ вычисляется в предлагаемой системе как:

$$MTTF_{system} = \left(\frac{MTTF_{node}}{MTTR * MPRC}\right)^{MPRC} * \frac{MTTF_{node}}{\prod_{i=0}^{MPRC}(|\boldsymbol{N}| - i)} \quad (21)$$

3. Результаты

В табл. 1 приведены расчеты надежности с использованием предложенной модели для DecStore по формуле (3), а также для известных подходов RAID-5, RAID-6 и системы с полным резервным копированием, где используется 1, 2 и 3 копии (FullBackup1, FullBackup2, FullBackup3 соответственно). Для расчета использовались следующие параметры: S = 2 TB;

Табл. 1

Вычисление R^2 для разных способов от 4 до 13 узлов.

Способ хранения	Количество узлов									
	4	5	6	7	8	9	10	11	12	13
DecStore 70%	99.99 9818 3437 3629 8815	99.99 9936 1363 3567 9631	99.99 9970 5715 6168 1875	99.99 9999 9998 8173 2588	99.99 9999 9999 5136 1520	99.99 9999 9999 7708 9316	99.99 9999 9999 9999 9944	99.99 9999 9999 9999 9976	99.99 9999 9999 9999 9988	99.99 9999 9999 9999 9994
DecStore 80%	89.70 0126 9276 8959 4233	99.99 9936 1362 3988 4602	99.99 9970 5715 0871 0996	99.99 9984 1049 4340 9072	99.99 9990 4669 1527 5376	99.99 9993 8416 4972 6632	99.99 9995 7949 2829 7730	99.99 9999 9999 9330 9326	99.99 9999 9999 9601 1778	99.99 9999 9999 9750 4035
DecStore 90%	89.70 0120 7477 9541 4298	92.75 7887 6240 0793 6068	94.43 8050 1866 6666 6744	95.49 3784 5771 6049 4122	96.21 6350 4554 5837 3393	96.74 1031 8325 8928 5693	99.99 9995 7949 1568 2657	99.99 9997 0026 0780 5764	99.99 9997 7889 5409 4634	99.99 9998 3227 3119 2357
Резервное копирование – 3 копии (FullBackup3)	99.99 9999 9894 8997 5884	99.99 9999 9868 6246 9855	99.99 9999 9842 3496 3826	99.99 9999 9816 0745 7797	99.99 9999 9789 7995 1769	99.99 9999 9763 5244 5740	99.99 9999 9737 2493 9711	99.99 9999 9710 9743 3682	99.99 9999 9684 6992 7653	99.99 9999 9658 4242 1624
Резервное копирование – 2 копии (FullBackup2)	99.99 9867 5736 9614 5124	99.99 9834 4671 2018 1405	99.99 9801 3605 4421 7687	99.99 9768 2539 6825 3968	99.99 9735 1473 9229 0249	99.99 9702 0408 1632 6530	99.99 9668 9342 4036 2811	99.99 9635 8276 6439 9092	99.99 9602 7210 8843 5374	99.99 9569 6145 1247 1655
Резервное копирование – 1 копия (FullBackup1)	97.49 7142 8571 4285 7142	96.87 1428 5714 2857 1428	96.24 5714 2857 1428 5870	95.62	94.99 4285 7142 8571 4285	94.36 8571 4285 7142 8805	93.74 2857 1428 5714 2857	93.11 7142 8571 4285 7500	92.49 1428 5714 2857 1740	91.86 5714 2857 1428 5376
RAID-5	99.99 9801 3605 4421 7687	99.99 9668 9342 4036 2811	99.99 9503 4013 6054 4217	99.99 9304 7619 0476 1904	99.99 9073 0158 7301 5872	99.99 8808 1632 6530 6122	99.99 8510 2040 8163 2653	99.99 8179 1383 2199 5464	99.99 7814 9659 8639 4558	99.99 7417 6870 7482 9931
RAID-6	99.99 9999 9894 8997 5884	99.99 9999 9737 2493 9711	99.99 9999 9474 4987 9422	99.99 9999 9080 3728 8989	99.99 9999 8528 5966 2383	99.99 9999 7792 8949 3575	99.99 9999 6846 9927 6536	99.99 9999 5664 6150 5237	99.99 9999 4219 4867 3649	99.99 9999 2485 3327 5744

[2] R=100*(1-8760/MTTF)

R_{rate} = 50%; Th = 30MB/s. Количество узлов варьируется от 3 до 12.

Для DecStore. процент означает свободное место на узлах, т.к. от его количества зависит надежность.

На рис. 3 представлены вычисления R по количеству девяток после запятой (99,999999 – это 6 девяток после запятой) в относительном виде. Если R < 99%, то на графике эти значения выводятся равным 0.

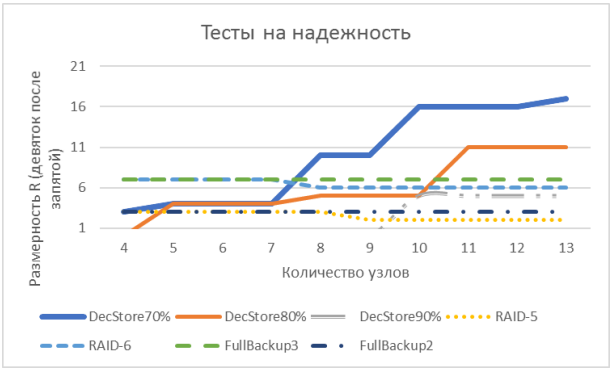

Рис. 3. Расчет надежности

Из графика видно, что подход к хранению файлов системы DecStore имеет меньшую надежность по сравнению с другими, когда число узлов мало. Но надежность DecStore повышается, когда система растет и увеличивает количество используемых серверов. Видно, что при увеличении количества узлов до 10 надежность DecStore становится выше. При этом надежность систем полного резервного копирования (FullBackup) становится независима от количества узлов после увеличения их количества. В случае традиционных технологий, таких как RAID-5 и RAID-6, с ростом количества узлов надежность становится меньше, т.к. вероятность выхода из строя двух или трех узлов одновременно увеличивается.

4. Процент сокращения хранения данных

Важно сформулировать правило, определяющее процент сокращения пространства при внедрении DecStore по сравнению с системами полного резервного копирования, который состоит из B резервных копий:

$$Storage\ reduction\ percentage = 100 * \left(1 - \frac{\frac{3}{2}}{1+B}\right)\% . \quad (22)$$

По сравнению с одним сервером резервного копирования требуемое пространство будет уменьшено на:

$$Storage\ reduction\ percentage = 100 * \left(1 - \frac{\frac{3}{2}}{1+1}\right)\% = 25\% . \quad (23)$$

Следовательно, предлагаемая система использует на 25% меньше места чем системы с полным резервным копированием, а по сравнению с системами, где для каждого узла используется 2 резервных (FullBackup2) – на 50%.

Заключение

В данной работе подробно описана математическая модель расчета надежности системы распределенного хранения файлов с использованием блокчейн-технологии и балансировщика нагрузки на базе смарт-контрактов. Произведенный расчет наглядно показывает, в чем преимущество такой системы и каким образом ее лучше применять. Данную модель можно применять не только для систем, построенных на блокчейн, но и для любых P2P, которые подразумевают способы автоматического восстановления данных на узлах

Наши выводы частично подтверждаются походим исследованием [16]. Но в нем показано сравнение для больших временных интервалов и больших дисков только для трех известных методов (двойное, тройное резервирование и RAID 5). В [17] обсуждается только проблема доступности узлов и ее зависимость от количества и способа репликации без учета техники восстановления данных типа erasure coding. Описанный здесь метод расчета надежности основан на выводах изложенных в [6]. В нашем исследовании мы пошли дальше и ввели новые параметры для расчета надежности, такие как время автоматического восстановления данных на других дисках и максимально возможное количество восстановлений после потери (*MPRC*).

Нужно отметить, что в DecStore системные узлы не могут быть использованы полностью. Нагрузка на узлы не должна превышать 80%, чтобы в системе было достаточно места для перестроения блоков данных. Однако в большинстве случаев рекомендуется, чтобы серверы имели меньшую загрузку для проблем, связанных с производительностью. Например, диски SSD работают лучше всего, если размер данных составляет менее 70%.

Представленные в Разделе 3 расчеты показывают, что система на базе блокчейн с балансировщиком нагрузки на основе смарт-контрактов может считаться более надежной, чем другие системы при увеличении количества узлов. Система лучше всего работает в сценариях, где требуется много узлов хранения (в примере более 7), а загрузка системы – не более 80%.

Литература

1. *Hammoud O., Tarkhanov I., Kosmarski A.* An Architecture for Distributed Electronic Documents Storage in Decentralized Blockchain B2B Applications // Computers. 2021. T. 10. № 11. C. 142.
2. *Хаммуд О., Тарханов И.А.* Методология хранения электронных документов в децентрализованных блокчейн приложениях (DApps) // Труды ИСА РАН. 2021.
3. *Hammoud O., Tarkhanov I.A.* A Novel Blockchain-Integrated Distributed Data Storage Model with Built-in Load Balancing // 2022 IEEE 16th International Conference on Application of Information and Communication Technologies (AICT). 2022.
4. *Wang X., Su J.* Research of Distributed Data Store Based on HDFS // 2013 International Conference on Computational and Information Sciences. 2013.
5. *Braam P.J.* Lustre: A Scalable, High-Performance File System. P. 13.
6. *Patterson D.A., Gibson G., Katz R.H.* A case for redundant arrays of inexpensive disks (RAID) // Proceedings of the 1988 ACM SIGMOD international conference on Management of data - SIGMOD '88. 1988.
7. *Kruglik S., Frolov A.* An information-theoretic approach for reliable distributed storage systems // Journal of Communications Technology and Electronics. 2020. T. 65. No 12. P. 1505–1516.
8. *Jia H. et al.* Reliability Analysis of distributed storage systems considering data loss and theft // Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability. 2019. T. 234. No 2. P. 303–321.
9. *Menčík J.* Reliability of Systems // Concise Reliability for Engineers. 2016.
10. *Balaji S.B. et al.* Erasure coding for distributed storage: An overview // Science China Information Sciences. 2018. T. 61. No 10.
11. Smart contracts and chaincode [Электронный ресурс] // hyperledger. URL: https://hyperledger-fabric.readthedocs.io/en/latest/smartcontract/smartcontract.html (дата обращения: 25.05.2023).
12. *Kannengieser N. et al.* Challenges and common solutions in smart contract development // IEEE Transactions on Software Engineering. 2022. T. 48. No 11. P. 4291- 4318.
13. *Arslan S.S.* Durability and Availability of Erasure-Coded Storage Systems with Concurrent Maintenance. // Notes on reliability theory. 2023. P. 4.
14. MTTF – What hard drive reliability really means [Электронный ресурс] // Toshiba. URL: https://www.toshiba-storage.com/trends-technology/mttf-what-hard-drive-reliability-really-means (дата обращения: 01.07.2023).
15. MQ04AB series – Toshiba Electronic Devices & Storage Corporation [Электронный ресурс] // Toshiba. URL: https://toshiba.semicon-storage.com/content/dam/toshiba-ss-v3/master/en/storage/product/internal-specialty/cHDD-MQ04AB_product-overview_r2s.pdf (дата обращения: 25.05.2023).
16. *Qin Xin et al.* Reliability mechanisms for very large storage systems // 20th IEEE/11th NASA Goddard Conference on Mass Storage Systems and Technologies, 2003. (MSST 2003). Proceedings.
17. *Ford et al.* Availability in Globally Distributed Storage Systems. Proceedings of OSDI. 2010. P. 61-74.

Тарханов Иван Александрович. Федеральное государственное учреждение «Федеральный исследовательский центр «Информатика и управление» Российской академии наук» г. Москва, Россия. Старший научный сотрудник. Кандидат технических наук, доцент. Область научных интересов: электронный документооборот, блокчейн, информационная безопасность. E-mail: tarkhanov@isa.ru (Ответственный за переписку).

Хаммуд Обадах. Национальный Исследовательский Технологический Университет (НИТУ) «МИСиС» Москва, Россия Научный ассистент. Область научных интересов: Dapps, блокчейн.
E-mail: obadah.hammoud@gmail.com

Analysis of the reliability of a distributed file storage system based on blockchain

O. Hammoud[I], I.A. Tarkhanov[II,III]

[I] National University of Science and Technology “MISiS”, Moscow, Russia

[II] State Academic University for Humanities, Moscow, Russia

[III] Federal Research Center “Computer Science and Control” of Russian Academy of Sciences, Moscow, Russia

Abstract. This paper assesses the reliability of the file storage system described in a series of articles [1,2,3]. This distributed system uses blockchain as a decentralized load balancer and distributes files across nodes using virtual disks. At the same time, the total amount of space required to ensure the reliability of the entire system is less than in full backup systems by 25%. However, it is impossible to calculate the reliability of such a system by known methods, since they do not take into account the time for automatic recovery of data on other nodes and the maximum number of recoveries possible after the loss of one node. The authors propose a mathematical method for calculating the reliability of such a class of systems. The article presents the result of the calculation by the presented method and, based on the results obtained, a conclusion is made about the advantages of load balancing systems based on smart contracts.

Keyword: *blockchain, p2p, distributed file system, reliability, load balancer.*
DOI: 10.14357/20790279230402 **EDN: YLCMKG**

References

1. *Hammoud O., Tarkhanov I., Kosmarski A.* An Architecture for Distributed Electronic Documents Storage in Decentralized Blockchain B2B Applications // Computers. 2021. T. 10. No 11. P. 142.
2. *Hammoud O., Tarkhanov I.A.* Methodology for storing electronic documents in decentralized applications on the blockchain (DApps) // Trudy Instituta sistemnogo analiza Rossiyskoy akademii nauk. 2021.
3. *Hammoud O., Tarkhanov I.A.* A Novel Blockchain-Integrated Distributed Data Storage Model with Built-in Load Balancing // 2022 IEEE 16th International Conference on Application of Information and Communication Technologies (AICT). 2022.
4. *Wang X., Su J.* Research of Distributed Data Store Based on HDFS // 2013 International Conference on Computational and Information Sciences. 2013.
5. *Braam P.J.* Lustre: A Scalable, High-Performance File System. P. 13.
6. *Patterson D.A., Gibson G., Katz R.H.* A case for redundant arrays of inexpensive disks (RAID) // Proceedings of the 1988 ACM SIGMOD international conference on Management of data - SIGMOD ’88. 1988.
7. *Kruglik S., Frolov A.* An information-theoretic approach for reliable distributed storage systems // Journal of Communications Technology and Electronics. 2020. Vol. 65. No 12. P. 1505–1516.
8. *Jia H. et al.* Reliability Analysis of distributed storage systems considering data loss and theft // Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability. 2019. Vol. 234. No 2. P. 303–321.
9. *Menčík J.* Reliability of Systems // Concise Reliability for Engineers. 2016.
10. *Balaji S.B. et al.* Erasure coding for distributed storage: An overview // Science China Information Sciences. 2018. Vol. 61. No 10.
11. Smart contracts and chaincode [Electronic resource] // hyperledger. URL: https://hyperledger-fabric.readthedocs.io/en/latest/smartcontract/smartcontract.html (accessed: 25.05.2023).
12. *Kannengieser N. et al.* Challenges and common solutions in smart contract development // IEEE Transactions on Software Engineering. 2022. Vol. 48. No 11. P. 4291-4318.
13. *Arslan S.S.* Durability and Availability of Erasure-Coded Storage Systems with Concurrent Maintenance. // Notes on reliability theory. 2023. P. 4.
14. MTTF – What hard drive reliability really means [Electronic resource] // Toshiba. URL: https://www.toshiba-storage.com/trends-technology/mttf-what-hard-drive-reliability-really-means (дата accessed: 01.07.2023).
15. MQ04AB series – Toshiba Electronic Devices & Storage Corporation [Electronic resource] // Toshiba. URL: https://toshiba.semicon-storage.com/content/dam/toshiba-ss-v3/master/en/storage/product/internal-specialty/cHDD-MQ04AB_product-overview_r2s.pdf (accessed: 25.05.2023).

16. *Qin Xin et al.* Reliability mechanisms for very large storage systems // 20th IEEE/11th NASA Goddard Conference on Mass Storage Systems and Technologies, 2003. (MSST 2003). Proceedings.
17. *Ford et al.* Availability in Globally Distributed Storage Systems. Proceedings of OSDI. 2010. P. 61-74.

Tarkhanov I.A. PhD, State Academic University for Humanities, Moscow, Russia, Moscow, Russia.
E-mail: tarkhanov@isa.ru

Hammoud Obadah. Assistant, National University of Science and Technology “MISiS”, Moscow, Russia.
E-mail: obadah.hammoud@gmail.com