Санкт-Петербургский государственный университет

Математическое обеспечение и администрирование информационных систем
Системное программирование

Муравьева Ольга Владимировна

Обработка и анализ результатов нагрузочного тестирования реализаций технологии Blockchain

Курсовая работа

Научный руководитель:
ст. преп. Я. А. Кириленко

Санкт-Петербург
2017

Оглавление

1. Введение

1.1. Blockchain

В последние годы начала набирать популярность новая технология под названием блокчейн. Блокчейн — это распределенная база данных. Впервые она была использована Сатоши Накомото в системе Bitcoin [1], как решение проблемы по созданию одновременно безопасной и не требующей администрирования системы связанных между собой записей.

Bitcoin быстро стала популярной, а к технологии блокчейн обратили свое внимание многие компании, благодаря чему появилось множество различных реализаций в чем-то сильно похожих, а в чем-то полностью отличающихся от оригинала. Существует большое количество областей, в которых может применяться эта технология, поэтому возникает необходимость оценки и сравнения различных реализаций.

Распределенные базы данных существовали и раньше. Основным нововведением блокчейн следует считать ее децентрализованность, то есть избавление от необходимости доверия одному центральному узлу.

База данных блокчейн поддерживает список, состоящий из записей о событиях в цифровом мире, который постоянно пополняется. Эти записи называют блоками. Каждый новый блок добавляется в систему лишь с согласия большинства участников, а после того, как информация записана изменить или удалить ее невозможно. В блоке содержатся сведения о времени его создания, ссылка на предыдущий блок, а также полный список входящих в него транзакций. Такая организация дает возможность объединить всю информацию о транзакциях проведенных с момента создания цепочки блоков. Цепочка целиком копируется на каждый узел системы, поэтому каждый участник получает достоверные сведения обо всех совершенных транзакциях.

Блокчейн продолжает существовать до тех пор, пока создаются новые блоки, независимо от количества добавления и удаления узлов в систему, а отсутствие какого-либо администрирующего узла позволя-

ет системе быть практически на сто процентов устойчивой к подделке данных.

1.2. Нагрузочное тестирование

Нагрузочное тестирование — это подвид тестирования на производительность, при котором на основе собранных технических показателей во время работы системы, оценивается ее ключевые показатели, такие как производительность, время отклика на внешние запросы.

Нагрузочное тестирование может проводиться для различных целей, например, определение соответствия системы требованиям, поиск узких мест в системе или сравнение производительности двух конкурирующих систем.

Для выполнения нагрузочного тестирования важно определить ключевые сценарии использования, включающие в себя наиболее распространенные сценарии, сценарии, сильно влияющие на производительность, важные бизнес-кейсы.

Также осуществляется выбор метрик и профилей нагрузки — набор запросов и временные рамки для различных сценариев использования системы.

Основные метрики нагрузочного тестирования:

- производительность — количество транзакций (запросов), обрабатываемых системой за единицу времени
- задержка — время распространения информации по системе
- время обработки — сколько времени нужно одному узлу для ответа на запрос другого
- использование ресурсов
  - центральный процессор
  - память
  - диск
  - сеть

1.3. О задаче

Из-за большого количества различных реализаций блокчейн встала задача эффективно сравнить их между собой, то есть создать систему способную провести испытания нагрузочного тестирования на данной технологии.

2. Решения из смежных областей

2.1. Распределенные базы данных

Yahoo Cloud Serving Benchmark — это проект компании Yahoo! для оценки производительности различных распределенных систем, таких как HBase, Cassandra, Riak, MongoDB и многих других. Проект состоит из двух частей: клиент для генерации изменяемой нагрузки и набор нагрузочных сценариев, которые могут быть выполнены генератором нагрузки.

Хотя базовые сценарии нагрузки дают законченное представление о производительности системы, есть возможность определения новых пользовательских сценариев нагрузки для изучения разных аспектов системы или сценариев работы приложения, не покрытых базовыми нагрузками.

Этот фреймворк предназначен для бенчмарка многих систем и их сравнения. Например, можно установить различные системы на одинаковое оборудование и провести тесты с одинаковой нагрузкой для различных систем, чтобы увидеть в чем и какая система превосходит остальные.

В [2] описывается работа с YCSB, сравнение производительности различных систем, рассматриваются такие метрики:

- производительность
  - задержка (средняя и 90/95-процентилей) относительно производительности (в процентах от максимальной)
  - задержка в зависимости от разных типов нагрузки (R/RW/W)
  - производительность в зависимости от разных типов нагрузки
- масштабируемость
  - задержка выполнения операции в зависимости от количества узлов в системе

  - производительность (количество операций в секунду) в зависимости от количества узлов
  - требуемый общий объем дисков в зависимости от количества узлов

- масштабируемость во время работы системы
  - полное время добавления новых узлов в систему и стабилизации работы
  - задержка (до/после/во время) в зависимости от количества добавляемых узлов в систему

- использование ресурсов
  - задержка и производительность при различных конфигурациях хранилища

2.2. Тестирование веб-серверов

Нагрузочное тестирование часто используется для оценки производительности веб-серверов, существует большое количество инструментов созданных специально для этого. Нагрузочное тестирование веб-сервера обычно проводится при сдаче его в повседневную эксплуатацию. При тестировании создаются условия, приближенные к реальности, чтобы оценить, достаточна ли мощность системы, правильно ли настроены приложения, участвующие в создании веб-контента, и прочие факторы, влияющие на работу веб-сервера.

Различают такие варианты теста:

- нагрузочный (Load-testing) — определяется работоспособность системы при некоторой строго заданной заранее (планируемой, рабочей) нагрузке.

- устойчивости (Stress) — применяется для проверки параметров системы в аномальных и экстремальных условиях, основная задача во время этого теста — попытаться нарушить работу систе-

мы. Позволяет определить минимально необходимые величины системных ресурсов для работы приложения, оценить предельные возможности системы и факторы, ограничивающие эти возможности. Также определяется способность системы к сохранению целостности данных при возникновении внештатных аварийных ситуаций.

Результат теста — максимальное число пользователей, которые могут одновременно получить доступ к веб-узлу, число запросов, обрабатываемых приложением, или время ответа сервера. Основываясь на полученном результате, можно заранее выявить узкие места, возникающие из-за несбалансированной работы компонентов, и исправить ситуацию, перед тем как включать систему в реальную работу.

При тестировании выдаются графики трех видов: линейный, нелинейный и насыщение. Отображающие соответственно, что сначала при возрастании нагрузки время отклика (т.е. обработки) остается постоянным, при дальнейшем увеличении нагрузки время отклика также увеличивается (почти линейно), и, наконец, наступает ситуация, подобная DOS-атаке, когда время отклика бесконечно увеличивается.

2.2.1. Jmeter

На основе [3] и [4] был выбран Apache Jmeter как один из наиболее популярных и удобных инструментов для автоматизации нагрузочного тестирования. Была предпринята попытка использовать его для выполнения тестовых сценариев, генерации нагрузки и сбора результатов.

Работа данного инструмента может быть описана так:

- создание тестового сценария, включающего в себя группы потоков (ThreadGroup)
- включение в каждую группу потоков элементов, отвечающих за отправку различных запросы (Sampler), и элементов, задающих логику отправки этих запросов (Controller)
- запуск готового сценария

- сохранение всех данных о работе сценария в специальном файле
- представление результатов в виде набора графиков

.

После более подробного рассмотрения данного инструмента мы пришли к выводу, что его нецелесообразно использовать в нашем проекте в силу таких причин:

- JMeter нацелен на то, чтобы отслеживать изменение состояния одного сервера под воздействием внешних запросов, нас же интересует поведение всей системы в целом
- со стороны генерации нагрузки получится использовать лишь малую часть функциональности данного инструмента, самостоятельная реализация требуемой функциональности проще, чем подгонка работы Jmeter под нужды нашего проекта

Как можно заметить, нагрузочное тестирование веб-серверов во многом сильно отличается от нагрузочного тестирования распределенных систем. В первом случае в приоритете находится оценка работы сервера при увеличивающемся количестве пользователей серверов и количестве поступающих от них запросов. А в случае с распределенной системой на первый план выходит оценка совместной производительности всех имеющихся узлов. Поэтому оказалось нецелесообразным использование существующих наработок по тестированию веб-серверов.

2.3. PeerUnit

Фреймворк [5] нацелен на тестирование централизованных точка-точка систем. На первый план они выдвигают оценку волатильности системы, то есть того, насколько легко она приспосабливается к изменениям(добавлению и удалению узлов). Также они акцентируют внимание на необходимости оценивать волатильность совместно с функциональностью системы. Третьим важным фактором они считают масштабируемость, которую также следует оценивать совместно с функциональностью.

Реализуется оценка посредством четырех шагов:

- мелкомасштабная неизменяющаяся система
- мелкомасштабная изменяющаяся система
- крупномасштабная неизменяющаяся система
- крупномасштабная изменяющаяся система

Данные шаги позволяют последовательно оценить:

- функциональность
- волатильность
- масштабируемость
- все аспекты в совокупности

Использовать данный фреймворк в нашем проекте невозможно, так как он нацелен на централизованные системы, но можно использовать их идею разделения тестовых сценариев по необходимым для оценки аспектам. Отмечу также, что в нашем случае на первое место выходят такие параметры как производительность и масштабируемость. В случае блокчейн волатильность системы не играет большой роли, потому что любая блокчейн продолжает нормально функционировать независимо от добавления или удаления узлов, а каждый узел содержит всю необходимую ему для работы информацию.

3. Постановка задачи

Задачей проекта являлось создание системы способной провести сравнительный анализ различных реализаций блокчейн. Моей основной задачей в рамках этого проекта являлось создание модуля пост-обработки данных, полученных в результате запуска блокчейна с различными сценариями нагрузки.

В рамках данной задачи следовало выполнить следующие подзадачи:

- изучение предметной области
  - изучение работы технологии блокчейн и понятия нагрузочного тестирования
  - изучение существующих решений в области нагрузочного тестирования распределенных систем
- создание основных сценариев нагрузки для тестирования
- определение той информации, которая должна быть получена от тестируемой системы и разработка формата в котором, она должна быть предоставлена
- выбор основных метрик рассчитываемых в результате исполнения каждого сценария, а также разработка формата в котором результат будет предоставляться на выход
- реализация модуля пост-обработки данных

4. Реализация

4.1. Выбор формата логов

Одним из первых возникших вопросов был выбор того, какие данные должны быть залогированы в результате работы тестирующей системы, а также в каком формате наиболее удобно представить имеющуюся информацию.

Все получаемые сведения могут быть разделены на такие части:

- информация со стороны модуля, генерирующего нагрузку
- информация от самих узлов блокчейн
- информация от вычислительной системы в облаке об используемых ресурсах

Форматом логов был выбран формат CSV.

Данные генератора нагрузки должны содержать информацию о каждой транзакции, отправляемой в систему, а также детали данной транзакции: ее идентификатор, время отправки в систему, размер и сообщение об успешности передачи транзакции. Каждый узел имеет в системе свой собственный модуль нагрузки, который пишет логируемые данные в соответствующий ему файл.

После тщательного анализа устройства блокчейн, было выяснено, что для получения полного представления о структуре цепочки блоков на каждом из узлов и ее изменении в течении теста, достаточно получить: во-первых, информацию о том, в каком блоке содержится каждая из транзакций, а во-вторых, данные о распространении блоков по узлам системы.

В результате для каждого узла генерируется файл с информацией о том, в какой момент времени была получена та или иная транзакция, а также файл содержащий сведения о нагруженности узла в каждый момент времени.

В то же время создается общий файл для сопоставления каждому существующему идентификатору транзакции, номера блока, содержащего ее.

В итоге полный формат логируемых файлов получился таким:

- nodeId-load.csv

  time, transactionId, transactionSize, responseMessage

- nodeId-blocks.csv

  blockId, arrivalTime

- transactionsPerBlock.csv

  transactionId, blockId

- nodeId-res-usage.csv

  time, cpu, usedMemory, usedMemory

4.2. Выбор ключевых сценариев

Второй подзадачей было определение набора тех сценариев, которые наиболее точно смогут продемонстрировать изменение ключевых метрик в зависимости от изменения конфигурации системы и интенсивности нагрузки.

Для получения более точных результатов подобранный список сценариев должен быть отработан на системах с качественно различным количеством нод, чтобы оценить возможности ее масштабируемости.

Сам список сценариев получился таким:

- транзакции фиксированного размера при увеличивающейся интенсивности

- фиксированная интенсивность при увеличивающемся размере транзакций

- фиксированная интенсивность при увеличивающемся размере блоков

- увеличивающаяся интенсивность, размеры транзакций, размеры блоков

Реализована была только обработка тестов, отработавших в формате первых двух сценариев. То есть может быть отображено влияние на работу всей системы таких параметров тестирования как интенсивность генерации нагрузки и средний размер генерируемых транзакций, а также могут быть найдены те состояния системы, в которых достигается наилучшая производительность, и точки, где происходит отказ в работе узлов из-за их чрезмерной перегруженности.

4.3. Выбор метрик и формата результатов

4.3.1. Метрики

Основной подзадачей являлся выбор ключевых метрик, дающих наиболее полное представление о работе той или иной реализации блокчейн и позволяющих производить как сравнение различных реализаций, так и сравнение работы одной и той же реализации при изменении ключевых параметров нагрузки.

Наиболее значительной метрикой была выбрана производительность (или пропускная способность) распределенной системы блокчейн, то есть то количество транзакций в секунду, которое данная система может интегрировать.

Считать транзакцию обработанной и добавленной в систему можно в различные моменты: например, когда модулем нагрузки был отправлен запрос на добавление транзакции, либо когда она была добавлена в блок, либо когда блок с ней распространился по определенному количеству узлов.

Был выбран последний вариант, так как рассматривать распределенную систему блокчейн следует только в совокупности всех имеющихся в ней узлов, а не изучая каждый узел по отдельности.

Другой значительной метрикой является задержка — скорость распространения блоков с транзакциями по узлам. В данном случае в ка-

честве задержки рассматривается время от создания блока до его распространения на 90% узлов блокчейна.

Для более подробного изучения поведения распределенной системы блокчейн были также выбраны такие дополнительные метрики:

- длина очереди транзакций — количество тех транзакций, которые уже были отправлены, но еще не были интегрированы в цепочку блоков
- количество генерируемых блоков за единицу времени
- интенсивность нагрузки
- размер блока
- размер транзакции
- задержка распределения транзакций по различному количеству узлов

Данные информация рассчитывается для каждого выполненного теста по получаемым от отработавшей системы лог-файлам.

4.3.2. Результирующие файлы

Также в рамках данной подзадачи следовало выбрать формат файлов для представления полученных метрик. Формат должен быть максимально удобным для того, чтобы любой желающий смог использовать его для отрисовки желаемых зависимостей.

В конечном итоге метрики представляются в таком формате пяти файлов:

- информацию о каждой из транзакций

  transactions.csv

  transactionId,blockId,transactionSize,

  transactionCreationTime,nodeId,blockLatency

- данные по всем блокам

  blocks.csv

  blockId,creationTime,blockLatency

- данные о задержках распространения транзакций по узлам

- данные о состоянии системы в каждый небольшой отрезок времени

  time.csv

  time,blockGeneration,throughput,latency,intensity,transactionSize,

  blockSize,numberTransactionsInBlock,transactionQueue

- использование ресурсов нагружаемой системой

  resources.csv

  time,nodeId,cpu,usedMemory,usedMemory

4.3.3. Визуализация

Собранные данные визуализируются. Во-первых, для каждого отдельного теста для каждой метрики отображается ее изменение с течением времени, а также ее гистограмма. Данные об используемых ресурсах отображаются для каждого узла отдельно.

Во-вторых, для групп тестов проведенных в рамках одного сценария строятся диаграммы размаха, объединяющие в себе данные о всех тестах и отображают изменение распределения ключевых метрик в рамках сценария. Отдельно отображается изменение среднего значения ключевых метрик от одного теста к другому в пределах сценария.

В-третьих, отображается сравнение средних значений ключевых метрик для различных реализаций блокчейн.

4.4. Детали реализации модуля обработки данных

Для реализации парсинга полученных логов был выбран язык программирования Java, модуль разделился на такие части как, разбор

существующих файлов, преобразование данных к требуемому виду и запись их в результирующие файлы.

Для отрисовки графиков из полученных логов был выбран язык R.

4.5. Тестирование реализаций блокчейн

Были проведены серии тестов и обработаны результаты для двух существующих реализаций блокчейн:

- Ethereum
- Hyperledger fabric

4.5.1. Ethereum

В результате первых попыток тестирования Ethereum были получены результаты не соответствующие ожиданиям: сильно скачущие метрики с течением времени (Рис. 1), отсутствие равномерности на графиках объединяющих результаты нескольких тестов в рамках сценариев (Рис. 2).

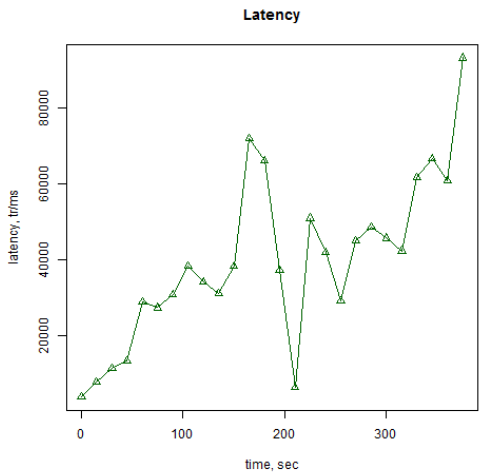

Рис. 1: Скачущее изменение задержки

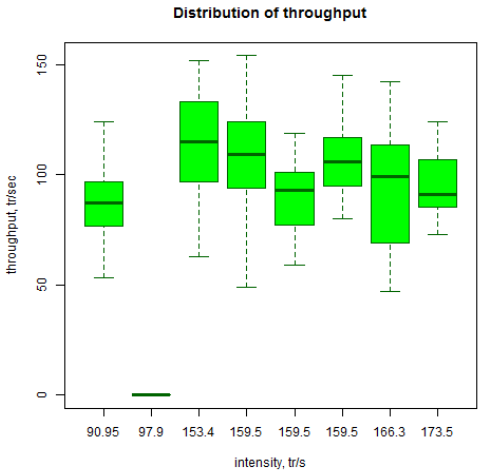

Рис. 2: Изменение пропускной способности для нескольких тестов

Проведение более длинных тестов(вместо 10 минутных, часовых) показало, что первые минуты теста являются непоказательными, так

как в это время работа блокчейна стабилизируется, следовательно производить сравнение и оценку на основе данных, собранных при коротких тестах, нецелесообразно. Также для сглаживания результирующих метрик была увеличена длина сегментов, на которые разбивается временная составляющая теста.

Некоторые проблемы этим были решены но не все. При проведении еще более длинного по времени теста(>4 часов) было замечено неожиданное поведение метрики интенсивности. Несмотря на то, что в каждом тесте нагрузка задается как постоянная величины, метрика интенсивности через какое-то время после начала теста начинала убывать (Рис. 3 и Рис. 4).

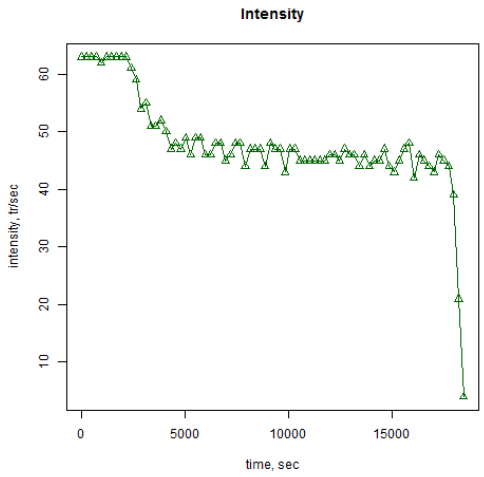

Рис. 3: Неравномерная нагрузка

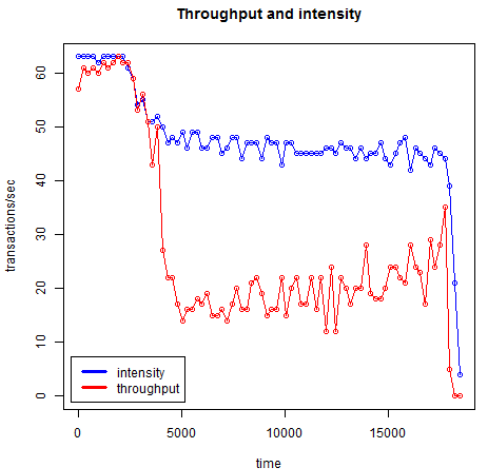

Рис. 4: Нагрузка и пропускная способность

Было определено, что данное поведение исходит из факта перегруженности блокчейн, в результате чего узлы блокчейн стопорят работу нагружающего узла, чем меняют интенсивность нагрузки. Были проведены тесты с меньшей нагрузкой, что позволило выровнять интенсивность нагрузки, и сопоставлены результаты для Ethereum в рамках 2-ух сценариев:

- изменяющейся интенсивности

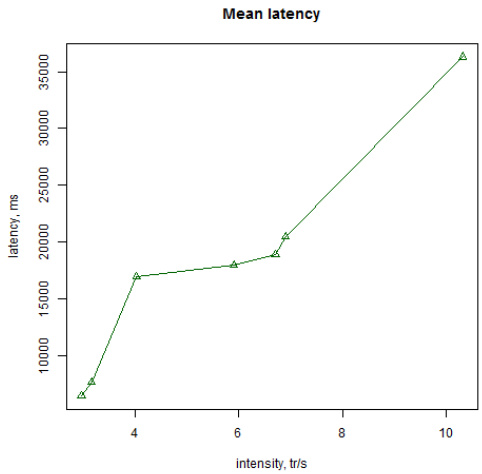

Рис. 5: Задержка

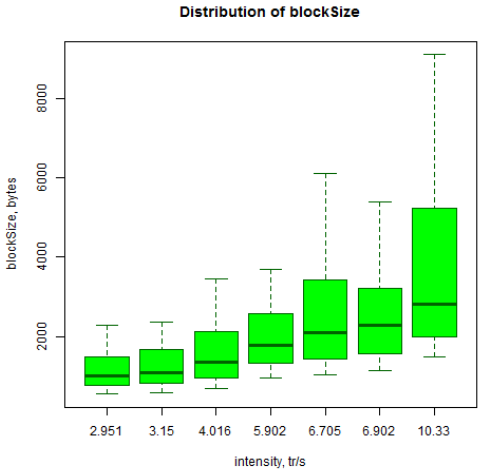

Рис. 6: Распределение размеров блока

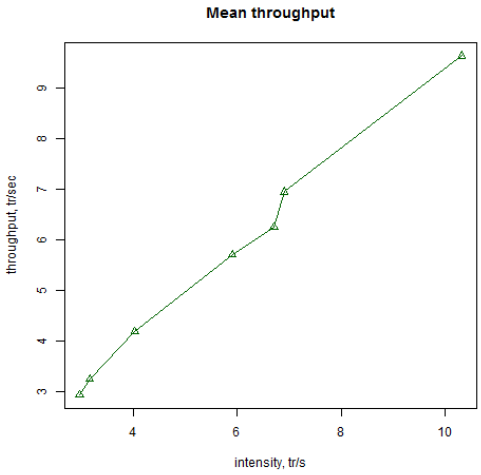

Рис. 7: Пропускная способность

- изменяющегося размера транзакций

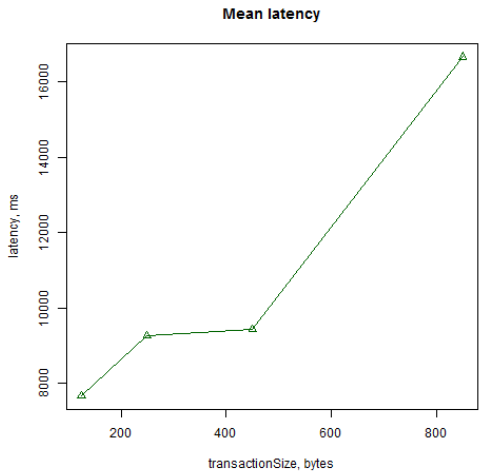

Рис. 8: Задержка

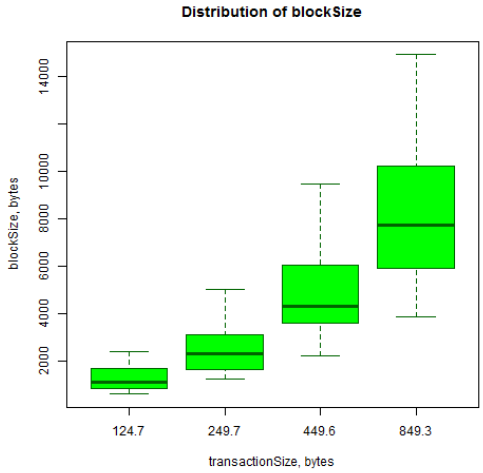

Рис. 9: Распределение размеров блока

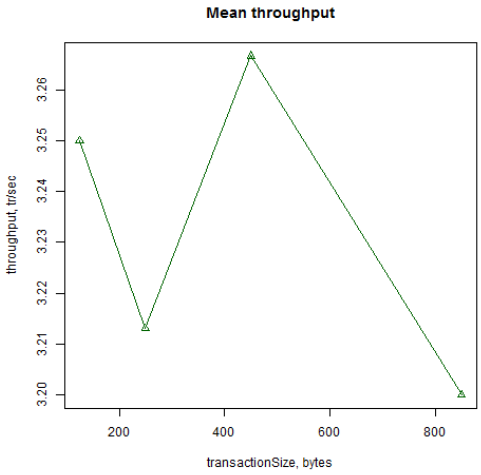

Рис. 10: Пропускная способность

4.5.2. Hyperledger fabric

Результаты первых тестирований проведенных на реализации fabric показали быстрый прирост расходуемой оперативной памяти (Рис. 12), по исчерпанию которой работа блокчейна прекращалась, пропускная способность (Рис. 11) и скорость генерации новых блоков падали до нуля.

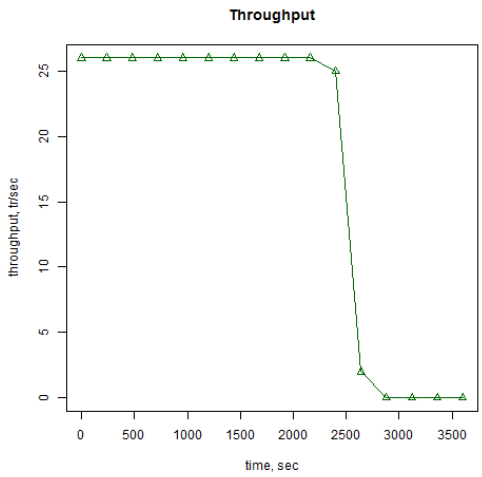

Рис. 11: Обрывание пропускной способности

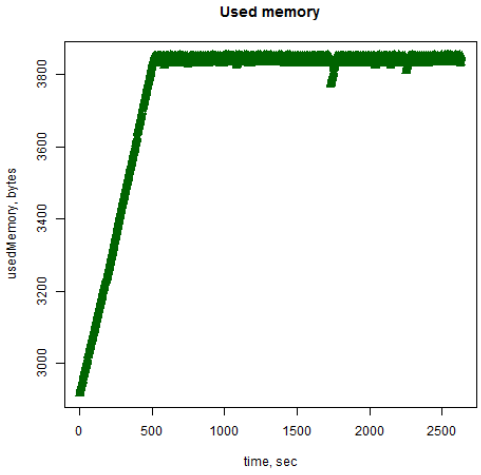

Рис. 12: Перегруженность оперативной памяти

Для решения данной проблемы была предпринята попытка заменить chaincode(основную работающую логику fabric) на бездействующую, то есть расходующую минимум оперативной памяти, это улучшило ситуацию и замедлило расход оперативной памяти, но проблему не решило.

Также была предпринята попытка запуска теста на узле с в два раза большим запасом оперативной памяти с целью найти тот предел, по достижению которого прирост расходуемой памяти заканчивается, но она оказалась неудачной и этот предел не был обнаружен.

Проблема так и не была решена, но при более низких нагрузках исчерпание памяти происходит не так быстро, поэтому можно рассмотреть результаты, которые были получены для первых двух часов работы тестов по сценарию изменения нагрузки(Рис. 13, Рис. 14).

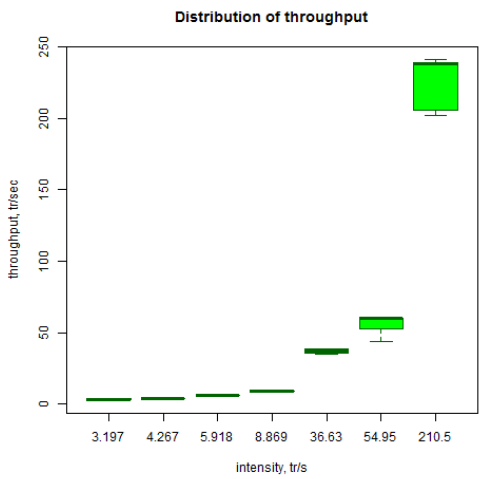

Рис. 13: Распределение пропускной способности

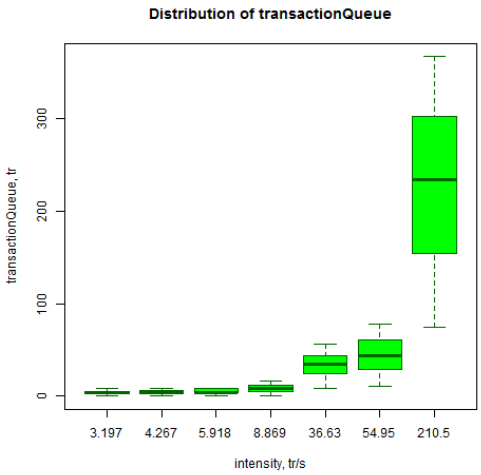

Рис. 14: Распределение очереди транзакций

4.5.3. Сравнение

Были получены также результаты сравнения для двух протестированных реализаций в рамках сценария изменяющейся интенсивности нагрузки.

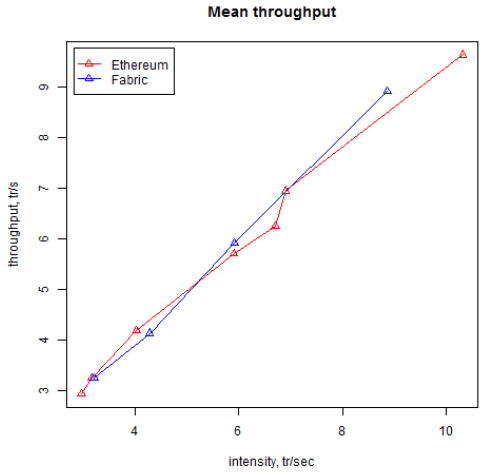

Рис. 15: Средняя пропускная способность

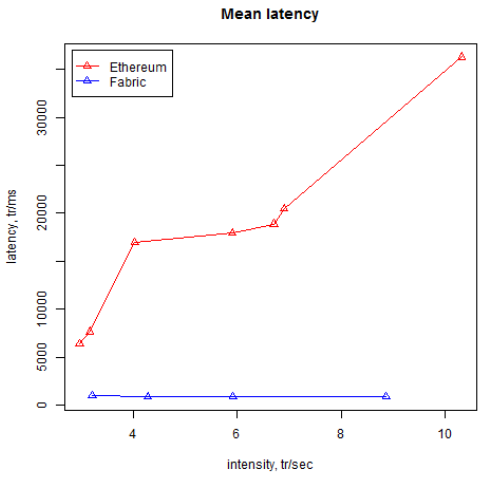

Рис. 16: Средняя задержка

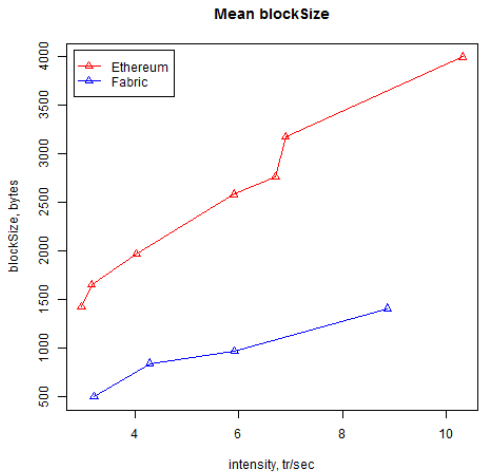

Рис. 17: Средний размер блока

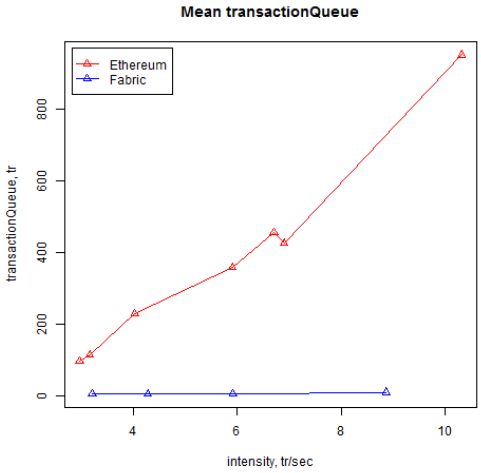

Рис. 18: Средняя очередь транзакций

5. Результаты

В конечном итоге был продуман и реализован модуль пост-обработки результатов для системы нагрузочного тестирования различных реализаций блокчейн. Данный модуль дает возможность сравнить такие ключевые параметры, как производительность, задержка и масштабируемость.

Сравнение возможно как для различных реализаций блокчейн, так и в условиях одной реализации при изменяющихся конфигурациях системы и параметрах нагрузки.

Список литературы

[1] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[2] Jorn Kuhlenkamp, Markus Klems, and Oliver Ross. Benchmarking scalability and elasticity of distributed database systems, 2014.

[3] Top 4 performance testing tools comparison, 2016.

[4] Open source load testing tools: Which one should you use?, 2015.

[5] Eduardo Cunha de Almeida, Gerson Sunyé, Yves Le Traon, and Patrick Valduriez. A framework for testing peer-to-peer systems, 2008.

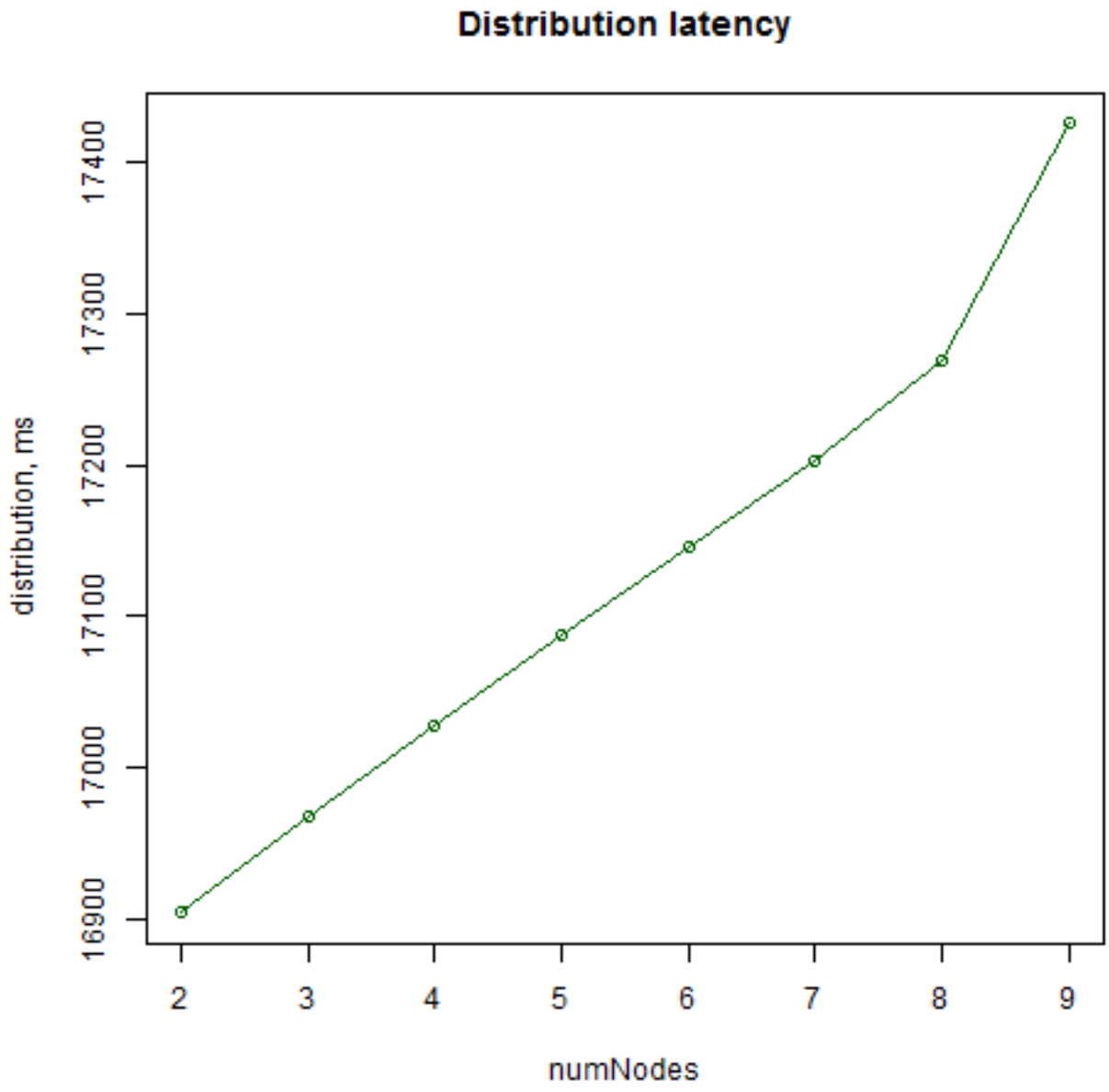

Рис. 19: Задержка распределения транзакций по узлам системы