ИНФОРМАЦИОННО-ИЗМЕРИТЕЛЬНЫЕ, УПРАВЛЯЮЩИЕ И СЕТЕВЫЕ СИСТЕМЫ

УДК 004.421.2

ПРИМЕНЕНИЕ АЛГОРИТМА PAGERANK ДЛЯ ЗАДАЧИ РАНЖИРОВАНИЯ В BFT СИСТЕМАХ

Э. К. Алгазинов, В. А. Музыченко

Воронежский государственный университет

Поступила в редакцию 26.08.2019 г.

Аннотация. Одной из тенденций в области распределенных систем является повышение интереса к таким распределенным системам, в которых узлы функционируют не только в ненадежной среде, но и в условиях отсутствия доверия между участниками самой системы. К таким, в частности, относятся системы, работающие на основе алгоритмов Byzantine Fault Tolerance (BFT). Эта тенденция имеет место и для поисковых систем. Стандартной задачей, решаемой поисковыми системами, является задача ранжирования результатов запросов. Для неё существуют несколько распространенных решений, которые применяются как в традиционных, так и в распределенных поисковых системах. Однако ориентация на BFT вводит ряд ограничений, которые затрудняют их применение. Алгоритм PageRank является одним из наиболее широко распространенных алгоритмов решения задачи ранжирования при организации поиска и рекомендаций в Web, задачах библиометрии, электронной коммерции, биологии и других. Он позволяет проводить оценку значимости объекта, основываясь на оценках его соседей, с которыми он имеет связи. Существуют исследования, посвященные адаптации данного алгоритма для некоторых типов распределенных систем. В рамках настоящего исследования решается проблема ранжирования результатов поиска в распределенной поисковой системе, работающей по BFT-алгоритму консенсуса. Предлагается вариант алгоритма ранжирования PageRank для распределенных BFT-систем, рассматриваются существующие модификации PageRank для распределенных систем, обсуждаются требования, предъявляемые к алгоритмам для работы в BFT-среде, исследуются параметры систем. В конце приводятся результаты эксперимента.

Ключевые слова: PageRank, BFT, ранжирование, поиск, распределенные системы, распределенные поисковые системы, распределенные вычисления, алгоритмы консенсуса.

ВВЕДЕНИЕ

Распределенные системы обладают рядом свойств, которые делают их интересными как с точки зрения решения прикладных задач, так и с исследовательской позиции. К таким свойствам можно отнести следующие: обеспечение повышенного уровня доступности системы, обеспечение сохранности данных, обеспечение устойчивости системы к ряду критических ошибок, связанных с отказом оборудования. Кроме того, данный тип систем имеет возможность горизонтального масштабирования: за счет добавления новых узлов можно добиться увеличения вычислительной мощности, объема памяти и других ресурсов системы.

Объектом данного исследования являются распределенные поисковые системы, которые используются для поиска информации на всем множестве web-сайтов и имеют широкое распространение в Web. Кроме того, такие системы применяются для организация поиска в больших объемах данных, когда нет возможности обработать все имеющиеся данные с помощью одной вычислительной машины, а также для поиска ресурсов в локальных сетях, в том числе, и в сетях интернета вещей (IoT).

Распределенные системы и распределенные поисковые системы, в частности, имеют ряд ограничений: зависимость от среды обмена сообщениями, сложность обеспечения целостности данных между узлами, необходимость принятие специальных мер для сохранения работоспособности при разделения системы на группы, а также обеспечение возможности восстановления целостности системы после устранения неполадок, вызвавших разделение.

Одной из тенденций в области распределенных систем является повышение интереса к таким системам, в которых узлы функционируют не только в ненадежной среде, но и в условиях отсутствия доверия между участниками самой системы. К таковым, в частности, относятся системы, в которых применяются алгоритмы семейства Byzantine Fault Tolerance (BFT).

BFT происходит от Задачи византийских генералов, заключающейся в выработке стратегии взаимодействия нескольких акторов для достижения некоторой цели, при условии, что часть из них может иметь скрытые мотивы, противоречащие основной цели.

Алгоритмы консенсуса – алгоритмы получения согласованного результата группой участников. Таким решением может быть добавление элементов в поисковый индекс, выполнение поисковых запросов и ранжирование результатов.

BFT алгоритмы консенсуса, в отличие от традиционных, позволяют принимать решения в условиях, когда узлы распределенной системы не могут в полной мере доверять друг другу.

К основным задачам, которые решаются поисковыми системами относятся следующие: обновление поискового индекса, поиск по индексу, ранжирование. Задача ранжирования заключается в сортировке результатов поисковой выдачи в порядке убывания релевантности.

Для решения задачи ранжирования результатов поиска часто применяется алгоритм PageRank [1, 2], который был впервые описан компанией Google и после публикации стал активно применяться во многих проектах, в том числе, не связанных с Web. Данный алгоритм, по сути, реализует применение к ранжированию web-страниц подхода, схожего с тем, который используется для определения индекса цитирования научных статей [1]. В его основе лежит идея, что информационная ценность документа (ранг) тем выше, чем больше на него ссылаются другие документы. Помимо количества ссылок так же важен и ранг ссылающихся документов. Иными словами, чем больше ссылок на документ и чем выше их ранг, тем выше и ранг документа. Таким образом, при выполнении ранжирования следует отдавать предпочтение документам, имеющим наибольшее количество входящих ссылок от других высокоранговых документов.

В рамках настоящего исследования решается задача ранжирования результатов поиска в распределенной поисковой системе, работающей по BFT-алгоритму консенсуса, с использованием алгоритма PageRank.

Актуальность комбинации данных алгоритмов обуславливается возрастающим интересом бизнеса к BFT-системам в общем и распределенным поисковым системам с BFT-консенсусом, в частности.

1. МАТЕРИАЛЫ И МЕТОДЫ ИССЛЕДОВАНИЯ

В настоящей работе используются следующие термины. Вершина – это вершина ориентированного графа, для которого необходимо выполнить ранжирование, например, графа веб страниц, связанных гиперссылками. Стоит отметить, что этот граф не является ациклическим. Узел – составная часть вычислительной системы, которая выполняет алгоритм ранжирования для графа документов.

1.1. Алгоритм PageRank

В основе алгоритма PageRank лежит модель посетителя, случайно переходящего по ссылкам (random surfer). Предполагается, что некоторый посетитель случайным образом переходит от одного документа к другому, используя связующие гиперссылки, и никогда

не возвращается назад при помощи соответствующей кнопки в браузере. Однако, иногда пользователь может перейти на случайную страницу, никак не связанную ссылками с последней.

Под рангом страницы будем подразумевать вероятность того, что посетитель окажется на данной странице. Так же введем параметр d – вероятность того, что пользователь перейдет на случайную страницу вместо перехода по ссылкам. Этот параметр может указываться как для всех страниц, так и для отдельных групп или индивидуальных страниц и управление им позволяет предотвратить умышленное манипулирования рангом страницы [1].

Пусть u – некоторый документ, F_u – множество документов, на которые ссылается документ u; B_u – множество документов, ссылающихся на u; $N_u = |F_u|$ – количество ссылок документа u на другие; $E(u)$ – вектор-параметр, предназначенный для документов и групп документов, не имеющих общих связей с другими; c – параметр, ограничивающий рост ранга [1].

Релевантность $R'(u)$ документа u в PageRank рассчитывается следующим образом [1, 2]:

$$R'(u) = c\left(\sum_{v \in B_u} \frac{R'(v)}{N_v} + E(u)\right), \qquad (1)$$

Таким образом, чем больше ссылок на страницу и чем выше их ранг, тем выше и ранг страницы. Параметр $E(u)$ необходим для того, чтобы релевантные документы, не имеющие связи с основным графом связей, могли все же получить достаточно высокий ранг и быть включенными в итоговую выдачу. Кроме того, за счет введения данного параметра решается проблема бесконечных ссылок. Граф связей между документами не обязательно должен быть ацикличным.

Классический алгоритм вычисление PageRank, описанный в [1, 2] приведен в листинге 1.

Здесь A – матрица с элементами a_{uv}, определяющая связи между документами, причем $A_{u,v} = 1 / N_u$ если документ u ссылается на v, и $a_{uv} = 0$, если такая связь отсутствует.

На первом шаге с помощью R_0 осуществляется начальная инициализация ранга страниц. Затем начинается главный цикл, на каждой итерации которого происходит вычисление более точного приближенного значения ранга страниц. Следующее приближение ранга страницы R_{i+1} вычисляется на основе предыдущего значения ранга R_i и матрицы связей A, а также параметра d, который рассчитывается как разница норм матриц R_i и R_{i+1}.

Листинг 1

Классический алгоритм расчета PageRank

$R_0 := S$
loop:
  $R_{i+1} := AR_i$
  $d := \| R_i \|_1 - \| R_{i+1} \|_1$
  $R_{i+1} := R_{i+1} + dE$
  $\delta := \| R_{i+1} - R_i \|_1$
while $\delta > \xi$

Работа алгоритма продолжается до тех пор, пока рассчитанные ранги страниц не стабилизируется. Другими словами, разница между следующим и предыдущим приближенными значениями ранга страниц не станет меньше заданной константы ξ.

1.2. Использование алгоритма в BFT-системе

Как отмечалось выше, изначально PageRank был разработан для системы Google, которая на тот момент, была централизованной системой [1]. С ростом популярности данный алгоритм стал применяться во многих реальных системах [3], в связи с чем возникло множество задач, связанных с его адаптацией для различных условий функционирования. Наша задача – сделать его пригодным для распределенных вычислений и децентрализованных систем. В оригинальных статьях [1, 2] не описывается его применение в распределенной среде.

Идея распределенных поисковых систем не нова, и по данной теме было проведено множество исследований [4–6]. Однако, теме использования алгоритма PageRank в BFT-системах не уделено достаточного внимания. К примеру решения, описанные в [4–5] не ориентированы на BFT-системы совсем, а проект [6], в рамках которого использование

данного алгоритма явно напрашивается, является коммерческим, и на текущий момент не известно ни публикаций, в которых бы раскрывалась данная тема, ни открытого исходного кода.

Наиболее известный и простой BFT-алгоритм, описанный Miguel Castro и Barbara Liskov в работе [7] и названный pBFT, был разработан для целей распределенной файловой системы. Благодаря его относительной простоте и практической эффективности, появление pBFT привело к широкому распространению идеи подобных алгоритмов. В своей работе авторы приводят результаты тестирования, согласно которым падение производительности системы после внедрения pBFT составило не более 3 % от нормы. Большинство современных BFT-алгоритмов так или иначе основываются на идеях, реализованных в pBFT.

В [7] было предложено соотношение, позволяющее рассчитать количество узлов BFT-системы, необходимое для успешного противодействия злонамеренным узлам:

$$N = 3f - 1, \qquad (2)$$

где N – общее число узлов, а f – максимально допустимое число потенциально злонамеренных узлов в системе.

Несмотря на то, что алгоритм pBFT был предложен много лет назад, в данной области он, в силу своей известности и простоты, является базовым и именно с ним проводят сравнения новых BFT-алгоритмов. В результате он является крайне удобным для проведения исследований. По этой причине в данной статье как основа используется именно он, однако полученные результаты могут быть перенесены и на другие BFT-алгоритмы,

В настоящей работе предлагается адаптация алгоритма ранжирования PageRank для распределенных BFT-систем.

В распределенных системах хранение данных может быть организовано в общем случае двумя способами: с помощью *репликации*, когда каждый узел хранит весь объем данных, или на основе *шардирования*, когда разные узлы могут хранить разные данные.

В случае, когда каждый узел имеет доступ ко всему объему данных, а также достаточно вычислительных ресурсов для их обработки, применение PageRank является тривиальным и никак не отличается от классического применения. Исключение представляет необходимость периодический синхронизации между узлами при пересчете рангов документов с целью сохранения целостности. Однако случай репликации по ряду причин встречается значительно реже шардирования. В одноранговых сетях документы, хранящиеся на одном узле, могут ссылаться на документы, хранящиеся на других узлах. В [8] предлагается алгоритм, в котором каждый узел выполняет предварительный расчет PageRank всех документов, хранящихся на нем, выделяет все ссылки на внешние документы и отправляет информацию об этом соответствующим узлам. Эта процедура продолжается для каждого документа до тех пор, пока его значение PageRank не стабилизируется. Другой вариант алгоритма был предложен в [9]. Здесь описывается применение модели вычислений MapReduce, заключающейся в разделении процесса на две части, известные как отображение (map) и свертка (reduce). В [10] приводится алгоритм с улучшенной эффективностью. В нем критикуется итеративный подход и предлагается подход на основе метода Монте-Карло. Утверждается, что, так как каждая следующая итерация требует результатов, полученных на предыдущей, то необходима постоянная синхронизация состояния и обмен сообщениями между узлами.

Метод Монте-Карло заключается в симуляции случайных переходов по графу и вычислении приближенного значения PageRank на этой основе. Сеть моделируется как граф $G = (V, E)$, где V – узлы (вершины) системы, E – связи. В начале каждый узел знает адреса только своих соседей в G. Подразумевается, что коммуникация проходит в рамках раундов. В качестве меры эффективности выбрано время, определяемое как количество раундов. Вычисления, выполняемые на узле и не подразумевающие коммуникацию, не учитываются.

Пусть ε – вероятность, с которой случайный переход начинается из конкретного случайно выбранного узла. Вектор PageRank гра-

фа G – вектор стационарного распределения π следующего случайного перехода: на каждом шаге случайного перехода с вероятностью ε переход начинается с некоторой вершины и с вероятностью $(1-\varepsilon)$ продолжается в некоторую другую вершину, связанную с данной.

2. РЕЗУЛЬТАТЫ И ИХ ОБСУЖДЕНИЕ

2.1. Шардирование в pBFT. Расчет конфигурации системы

Из приведенных выше рассуждений очевидно, что каждая часть графа должна храниться не менее чем одним узлом системы для обеспечения сохранности данных при отказах узлов, однако из-за необходимости обеспечения сохранности данных при BFT-ошибках это количество следует увеличить.

Так как f узлов системы могут оказаться злонамеренными и действовать скоординировано, а именно договориться предоставлять одни и те же ложные данные, то нужно как минимум $f+1$ корректных узлов, чтобы противостоять им, а, следовательно, n – общее число узлов системы не может быть меньше, чем $(2f+1)$. На практике часто используется соотношение $n-2f>f \Rightarrow n>3f$, исходя из того, что в некоторых случаях не все корректные узлы могут ответить [7].

Помимо общего числа узлов в системе необходимо рассчитать g – количество узлов в каждой отдельной группе, то есть хранящих одни и те же данные. Так же, как и общее число узлов, количество узлов в группе не может быть меньше, чем $(2f+1)$, то есть $g \geq 2f+1$. Однако, в этом случае есть и ограничение сверху: $g<n$. Случай $g>n$ невозможен. При $g=n$ в системе может быть только одна группа, что допустимо с точки зрения BFT, но приводит к тому, что каждый узел должен хранить все данные графа, а это сводит на нет некоторые плюсы применения распределенных вычислений.

Стоит учесть, что количество узлов в системе может меняться из-за сбоев на отдельных узлах, в случае потери связи между узлами и при некоторых других обстоятельствах, что неизбежно приводит к изменению коэффициента репликации, то есть количества узлов, хранящих один и тот же блок данных. Чтобы можно было восстановить необходимый уровень репликации после выхода r узлов из g, хранящих один и тот же блок, с учетом наличия максимум f «плохих» узлов, а также с соблюдением условия $g<n$, получаем, что $g=2f+r+1$.

Заметим, что общее количество групп G, которые могут существовать в системе при заданных и определяется формулой:

$$\frac{n!}{g!(n-g)!}, \qquad (3)$$

Используя полученные выше соотношения, опишем **алгоритм расчета PageRank в BFT-системе**. Предлагаемый алгоритм, основанный на идеях [10] с учетом описанных в настоящей работе особенностей BFT-систем, приводится в листингах 2 и 3. Он состоит из следующих шагов:

1. Весь процесс вычисления делится на раунды.

2. Узлы в каждом раунде выполняют серию случайных переходов по ссылкам, исходящим из всех вершин, хранящихся на них. Максимальное число случайных переходов из конкретной вершины регулируется параметром *couponCount*.

3. После выполнения необходимого количества переходов узлы одной группы обмениваются результатами с целью обобщения.

4. Мастер-узел посылает общий результат в другие группы, остальные узлы посылают группам хеш-сумму сообщения и криптографическую подпись.

5. Получив блок от других групп, узлы проверяют подписи и оповещают другие узлы своей группы.

6. Полученный блок случайных переходов, выполненных другими группами, используется для расчета PageRank локальных узлов.

7. Описанная последовательность повторяется для следующего раунда, пока список раундов не будет исчерпан.

Здесь T – матрица посещений узлов; i – идентификатор раунда; B – некоторая достаточно большая константа; $c=2/(\delta^{'}\varepsilon)$ и $\delta^{'}$ –

параметры, регулирующие величину ранга, K – максимально допустимое количество случайных переходов из каждой вершины на первой итерации алгоритма, ξ_v – количество переходов в вершину v, ε – вероятность перехода к случайной вершине, R_v – PageRank ранг документа v, Nodes – узлы поисковой системы. Процедура $exchangeWalks(T, i)$ – процедура обмена информацией о совершенных случайных переходах между узлами системы.

Каждый узел выполняет следующие шаги:

1. Инициализация начального значения *couponCount*.

2. Инициализация начального количества переходов ξ_v в вершину v из вершин-соседей, т. е. таких, что $v \in L_u$.

3. Выполнение случайных переходов.

4. Обмен информацией о случайных переходах, совершенных в раунде i с другими узлами (Листинг 3).

5. Расчет *couponCount* для следующего раунда.

6. Вычисление приближенного значение PageRank для вершины v.

В алгоритме, описанном в листинге 3, каждый узел рассылает другим узлам в своей группе список переходов, совершенных на узлы, хранящиеся за пределами группы. Так как переходы выполняются случайно, то разные узлы выполняют их по-разному. По этой причине результаты усредняются и согласовываются при помощи BFT-консенсуса. Мастер-узел отправляет сообщение с информацией о переходах в другие группы, а остальные узлы, в целях экономии ресурсов,

Листинг 2

Алгоритм расчета PageRank локальных документов в BFT системе

$couponCount_v := K$ для всех v
$\xi_v := K$
for раунд $i = 1, 2, 3, ..., B \lg \frac{n}{\varepsilon}$ **do**
 for $v \in Nodes$ **do**
  $T_v^u := 0$ для всех u, на которые v имеет прямые ссылки ($u \in L_v$)
 end
 for $v \in Nodes$ if $couponCount_v > 0$ **do**
  # Выполнение случайных переходов
  for $j = 0, 1, 2, ..., couponCount_v$ **do**
   $T_v^u := T_v^u + 1$ для случайного $u \in L_v$
  end
  # Обмен информацией о случайных переходах, совершенных в раунде i с другими узлами (Листинг 3)
  $\hat{T} := T$ для всех u, не хранящихся на данном узле
  $T' := exchangeWalks(\hat{T}, i)$
 end
 for $v \in Nodes$ **do**
  $\xi_u := \xi_u + \sum_{v \in N_u} T'^u_v$
  $couponCount_v := \sum_{v \in N_u} T'^u_v$
 end
end

for $v \in Nodes$ **do**
 # Вычисление приближенного значение PageRank для узла v
 $R_v := \frac{\xi_v \varepsilon}{cn \lg n}$
end

отправляют только хэш-сумму сообщения с подписью. После получения всех необходимых сообщений функция возвращает матрицу согласованных переходов.

Листинг 3

Алгоритм BFT уведомления других узлов

```
function exchangeWalks(T, i) do
 T^agreed := doBFT(T, i)
 if мастер узел do
   h := hash(T^agreed, i)
   notifyOthers(T^agreed, i, h)
 else do
   notifyOthers(None, i, h)
 end
 T' := waitOthers(i)
 notifyOwn(hash(T'), i)
 waitOwn(i)
 return T'
end
```

Здесь *notifyOthers* – процедура, реализующая отправку данных узлам в других группах; *waitOthers* – процедура ожидания получения информации о совершенных переходах от других групп; *notifyOwn* – процедура уведомления узлов, входящих в ту же группу, что и данный узел; *waitOwn* – процедура ожидания получения о совершенных переходах от узлов своей группы; *doBFT* – процедура выполнения BFT-консенсуса для синхронизации случайных переходов, совершенных разными узлами внутри группы; *hash* – процедура вычисления хеш-суммы; *None* – пустое значение.

2.2. Вычислительный эксперимент

Была проведена серия вычислительных экспериментов с предложенным в настоящей работе алгоритмом. Использовалась реализация на языке программирования Scala, данные хранились в оперативной памяти и в базе RocksDB. Все эксперименты были проведены в виртуальной среде с применением системы контейнеризации Docker. Эксперименты проводились для случая $f = 10$, количество узлов системы составляло 31 узел.

2.2.1. Исследование алгоритма с точки зрения получаемых результатов ранжирования

Для оценки релевантности была применена метрика NDCG – часто используемая метрика для оценки эффективности алгоритмов поисковых систем. Данная метрика дает оценку релевантности в промежутке [0, 1], при этом 1 соответствует идеальному случаю, когда документы в выдаче отсортированы точно в соответствии с ожиданием.

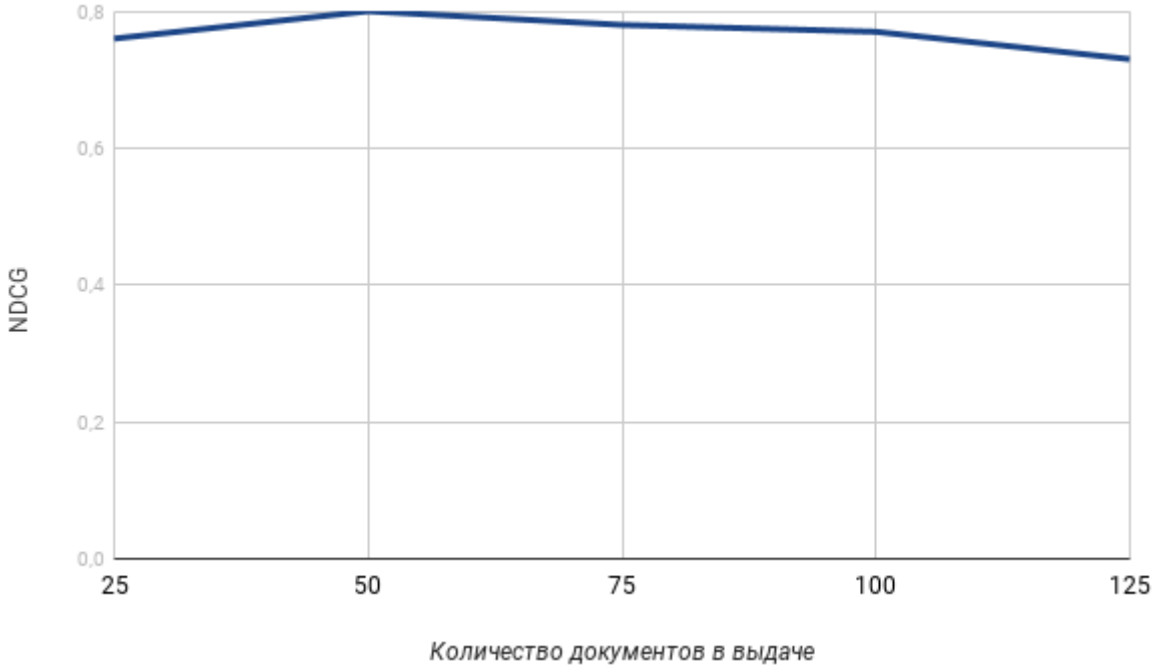

Рис. 1. Метрика NDCG в зависимости от количества документов в выдаче

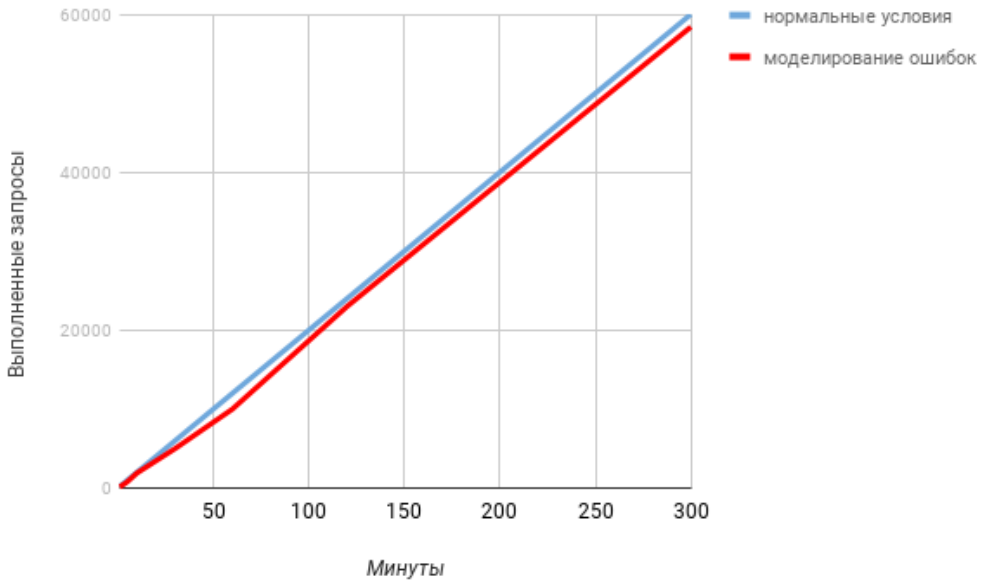

Рис 2. Сравнение показателей ранжирования в нормальных условиях и с ошибками

Для первых 25, 50, 75, 100 и 125 документов в выдаче была выполнена оценка. Результаты приведены на рис. 1. Можно заметить, что средняя оценка не опускается ниже 0.75, что является хорошим показателем.

2.2.2. Исследование производительности

В данном эксперименте проверялась разница в производительности системы в нормальных условиях и в случае возникновения ошибок. Применялась симуляция одновременно двух типов ошибок. Ошибки в сети были симулированы средствами Docker путем добавления правил, запрещающих обмен данными между узлами. Для BFT-ошибок были внесены изменения в код узлов, благодаря которым некоторые из них нарушали порядок обмена сообщениями, игнорируя или отправляя сообщения с заведомо неверными данными. Результаты приводятся на рис. 2.

Производительность системы в случае ошибок несколько меньше, чем в нормальных условиях, однако система продолжает функционировать. Такое поведение является нормальным, так как при увеличении количества ошибок возрастают и накладные расходы на противодействие им, в результате остается меньше ресурсов на выполнение основной задачи.

Принимая во внимание полученные результаты можно сделать вывод, что предложенный алгоритм успешно решает задачу ранжирования результатов поиска в условиях распределенной BFT-системы.

ЗАКЛЮЧЕНИЕ

В настоящей работе предложена адаптация алгоритма ранжирования PageRank для распределенных BFT-систем, рассмотрены существующие модификации PageRank для распределенных систем, требования, предъявляемые к алгоритмам для работы в BFT-среде.

Результаты и материалы статьи могут быть использованы для решения прикладных задач при реализации распределенных вычислительных систем, распределенных поисковых систем, распределенных систем хранения данных.

СПИСОК ЛИТЕРАТУРЫ

1. *Page, L.* The anatomy of a large-scale hypertextual Web search engine / L. Page, S. Brin // Computer Networks and ISDN Systems, Volume 30, Issues – 1998. – С. 107–117.

2. The PageRank Citation Ranking: Bringing Order to the Web / L. Page и др. // Stanford InfoLab – 1999. – С. 1–17. – Режим доступа: http://ilpubs.stanford.edu:8090/422/1/1999-66.pdf – (Дата обращения 01.11.2017).

3. *Gleich, D. F.* PageRank Beyond the Web / David F. Gleich // Society for Industrial and Applied Mathematics – 2015. – С. 321–363.

4. *Gravano, L.* The Efficacy of GlOSS for the Text Database Discovery Problem / L. Gravano, H. Garcia-Molina, A. Tomasic // Stanford InfoLab – 1993. – С. 1–39. – Режим доступа: http://ilpubs.stanford.edu:8090/29/1/1993-21.pdf – (Дата обращения 01.11.2017)

5. YaCy Decentralized Web Search. – Режим доступа: https://yacy.net/en/index.html – (Дата обращения 21.11.2017).

6. The Community-Powered Search Engine. – 2017. – С. 1–39. – Режим доступа: https://www.presearch.io/uploads/WhitePaper.pdf – (Дата обращения 21.11.2017)

7. *Castro, M.* Practical Byzantine Fault Tolerance / M. Castro, B. Liskov // Laboratory for Computer Science, Massachusetts Institute of Technology, 545 Technology Square, Cambridge. – 1999. – С. 1–14.

8. *Sankaralingam, K.* Distributed Pagerank for P2P Systems / K. Sankaralingam, S. Sethumadhavan, J. C. Browne // High Performance Distributed Computing, 2003. Proceedings. 12th IEEE International Symposium on High Performance Distributed Computing. – 2003. – С. 1–11.

9. *Bahmani, B.* Fast personalized pagerank on mapreduce / B. Bahmani, K. Chakrabarti, D. Xin // In Proc. of ACM SIGMOD Conference. – 2011. – С. 973-984.

10. *Sarma, A. D.* Fast Distributed PageRank Computation / A. D. Sarma и др.l // Theoretical Computer Science, volume 561. – 2015. – С. 113-121.

Алгазинов Эдуард Константинович – д-р физ.-мат. наук, проф., декан факультета компьютерных наук, Воронежский государственный университет. E-mail: algazinov@sc.vsu.ru

Музыченко В. А. – аспирант, кафедра программирования и информационных технологий, факультет компьютерных наук, Воронежский государственный университет.
E-mail: yau.ref@gmail.com

ON APPLYING PAGERANK ALGORITHM FOR RANKING IN BFT SYSTEMS

E. K. Algazinov, V. A. Muzychenko

Voronezh State University

Annotation. One of the modern trends in the field of distributed systems is an increasing interest in such distributed systems where nodes operate not only in an unreliable environment, but also in the absence of trust between participants in the system itself. These in particular include systems which utilize Byzantine Fault Tolerance (BFT) algorithms. This trend is also observed for search engines. One of the problems to be solved when developing search engines is ranking. There are several common solutions for this problem which are used both in traditional and in distributed search engines. However, BFT introduces a number of limitations that make it difficult to apply these solutions. The PageRank algorithm is one of the most widespread algorithms for solving the ranking problem when organizing searches and recommendations on the Web, problems of bibliometry, e-commerce, biology and some others. It allows you to assess the significance of the object, based on the estimates of its neighbors it has connections with. There are studies devoted to adapting this algorithm to ensure its applicability for distributed computing in some types of distributed systems. This study focuses on the problem of ranking in a distributed search system that works with BFT consensus algorithm by proposing a version of the PageRank ranking algorithm for distributed BFT systems. It includes overview of existing PageRank modifications for distributed system, as well as requirements for algorithms for working in a BFT environment. At the end there are the results of experiment.

Keywords: PageRank, BFT, webpage ranking, search, distributed systems, distributed search systems, distributed computing, consensus algorithms.

Algazinov Edward K. – Doctor of Physical and Mathematical Sciences, Professor, Dean of the Faculty of Computer Science, Voronezh State University. E-mail: algazinov@sc.vsu.ru

Muzychenko V. A. – graduate student of Computer science faculty of Voronezh State University.
E-mail: yau.ref@gmail.com