Распределённые алгоритмы

mk.cs.msu.ru → Лекционные курсы → Распределённые алгоритмы

Блок 43

Паксос

Лектор:
Подымов Владислав Васильевич
E-mail:
valdus@yandex.ru

ВМК МГУ, 2025, февраль–май

Вступление и общие особенности

Паксос — это семейство алгоритмов консенсуса над произвольным наперёд заданным множеством решений, широко применяющихся на практике, например, для

- согласования транзакций в распределённых базах данных,
- согласования истории блоков в блокчейне,
- согласованного доступа к файлам в распределённых файловых системах (например, в менеджере блокировок Google Chubby для Google File System) и
- в целом в тех случаях, когда требуется разумный и достаточно эффективный консенсус в сети, допускающей много «сильных» неисправностей

Ввиду известных результатов о невозможности консенсуса, придётся «пожертвовать» свойством завершаемости: алгоритм будет завершаться только если в сети достаточно много исправных узлов, достаточно быстро обменивающихся сообщениями

Вступление и общие особенности

Название «Паксос» (Paxos) можно считать бессмысленным: его предложил в 1998 году Л. Лэмпорт в первом описании этого алгоритма в рамках иллюстрации его как схемы голосования в вымышленном парламенте на (реально существующем) греческом острове Паксос

Обсудим этот вариант алгоритма (из статьи «Part-time parliament»), переизложенный понятнее в 2001 году (в статье «Paxos made simple»)

В этом алгоритме используется особая модель отказов:

1. Узлы могут выходить из строя и восстанавливаться с частичным восстановлением состояния на момент сбоя
   - *О том, какие именно части состояния восстанавливаются, будет рассказано позже*
2. Сообщения могут доставляться сколь угодно долго, теряться и дублироваться, но не могут искажаться
   - *То есть, в частности, нет полноценных византийских отказов*

Роли

Узлам сети раздаются роли в голосовании: заявитель, избиратель, наблюдатель

Одному узлу может быть присвоено и несколько ролей

Все узлы знают свои роли и роли всех других узлов

Заявитель выдвигает на голосование заявку, содержащую решение

Каждый избиратель одобряет или отклоняет заявку, и на основании этих действий совокупность всех избирателей (электорат) может выбрать заявку

Все узлы принимают решение, содержащееся в заявке, выбранной электоратом

T1 и выбор большинством голосов

Самый простой способ организовать выборы — это предоставить выбор одному избирателю p:

- Заявители доносят свои заявки до p
- Какую заявку p одобрил, такую электорат $\{p\}$ и выбирает

Так как требуется одобрить заявку даже в том случае, если она всего одна, то естественно возникает следующее требование к выборам:

T1. Избиратель должен одобрить первую полученную заявку

Но в такой схеме выборов выход из строя всего одного узла-избирателя приводит к тому, что никакое значение не будет выбрано

Чтобы преодолеть эту проблему, можно увеличить число избирателей:

1. Каждый избиратель может одобрить или отклонить заявку
2. Заявка выбирается только в том случае, если она одобрена большинством избирателей

T1 и выбор большинством голосов

Если каждый избиратель будет одобрять **только** первую полученную заявку, то выборы могут быть легко сорваны даже без отказов узлов и ошибок передачи сообщений — например:

- Электорату из пяти избирателей поступили заявки a, b и c
- Два избирателя первой получили (и одобрили) заявку a, два — заявку b и один — заявку c
- Ни для одной заявки не набрано большинство голосов

Чтобы преодолеть эту проблему, следует разрешить избирателю одобрять не только первую полученную заявку, но и другие

Но тогда избирателем может быть одобрено и электоратом выбрано несколько разных решений, а для консенсуса необходимо выбрать только одно

Нумерация заявок

Чтобы можно было отслеживать статус конкретных заявок (как с одинаковыми, так и с разными значениями), пронумеруем их: при выдвижении заявитель присваивает заявке номер так, чтобы

- различные заявки имели различные номера и
- номера заявок были линейно упорядочены

Например, если узлам сети присвоены уникальные идентификаторы из линейно упорядоченного множества, то

- номером заявки может служить пара $[p, n]$, где p — идентификатор заявителя и n — то, какую по счёту заявку он выдвигает, и
- линейно сравнивать такие номера заявок можно, например, лексикографически

Заявкой будем считать пару $(v, \mathfrak{n})$, состоящую из значения v (решения) и номера $\mathfrak{n}$ (элемента линейно упорядоченного множества)

T2, T2И

Тогда консенсус можно обеспечить, предъявив такое «глобальное» требование к электорату и заявителям:

T2. Если выбирается заявка $(v, \mathfrak{n})$, то заявка $(w, \mathfrak{m})$ для $\mathfrak{m} > \mathfrak{n}$ может быть выбрана только в том случае, если $w = v$

T2 не запрещает выбирать несколько заявок, если значения этих заявок одинаковы

Чтобы соблюсти T2, ограничим свободу действий избирателя так:

T2И. Если выбирается заявка $(v, \mathfrak{n})$, то заявка $(w, \mathfrak{m})$ для $\mathfrak{m} > \mathfrak{n}$ может быть одобрена избирателем только в том случае, если $w = v$

Утверждение. **Если для каждого избирателя верно T2И, то верно и T2**

T2З

Так как возможны потери сообщений, то T1 и T2И могут друг другу противоречить, если не ограничивать свободу выдвижения заявок — например:

1. Выдвигаются заявки $(a, 1)$ и $(b, 2)$ для электората $\{p, q, r\}$
2. Заявка $(a, 1)$ одобряется p и q и выбирается большинством голосов
3. После этого r получает заявку $(b, 2)$ и по T1 обязан её одобрить, а по T2И — отклонить

Чтобы таких противоречий не возникало, ограничим свободу действий заявителя так:

T2З. Если выбирается заявка $(v, \mathfrak{n})$, то заявка $(w, \mathfrak{m})$ для $\mathfrak{m} > \mathfrak{n}$ может быть выдвинута заявителем только в том случае, если $w = v$

Утверждение. **Если для каждого заявителя верно T2З, то для каждого избирателя верно T2И**

T2B

Чтобы T2З можно было соблюдать более «конструктивно», добавим следующее требование:

T2B. Заявка $(v, \mathfrak{n})$ может быть выдвинута только в том случае, если существует множество избирателей S, содержащее более половины избирателей и такое что

- либо ни один узел из S не одобряет заявки с номерами меньше $\mathfrak{n}$,
- либо для заявки $(w, \mathfrak{m})$, одобряемой кем-либо из S и такой что $\mathfrak{m} < \mathfrak{n}$, с наибольшим номером $\mathfrak{m}$ среди таких заявок, верно $w = v$

Утверждение (Д.з. 1). Если для каждого заявителя верно T2B, то для каждого заявителя верно и T2З

Но так как заявка с меньшим номером может быть выдвинута и одобрена избирателями хронологически позже заявки с бóльшим номером, то для соблюдения T2B может быть затруднительно (*и неэффективно*) отслеживать значение и получение такой «запаздывающей» заявки

Бронирование заявки

Вместо отслеживания статуса заявки можно добиться соблюдения T2B при помощи бронирования заявки:

- Заявитель отправляет всем избирателям номер $\mathfrak{n}$ заявки, которую хочет выдвинуть
- Избиратель, получив такой номер,
  - обещает больше не одобрять заявки с номерами, меньшими $\mathfrak{n}$, и
  - отправляет в ответ одобренную им заявку $(v, \mathfrak{m})$ с наибольшим номером $\mathfrak{m}$, таким что $\mathfrak{m} < \mathfrak{n}$, или сообщение об отсутствии таких заявок
- Если в ответ на бронирование заявитель получил хотя бы одну заявку, то перед рассылкой заявки он изменяет значение заявки на значение полученной заявки с наибольшим номером (*чтобы заявка следовала T2B*)

Утверждение. **Для каждого заявителя, бронирующего заявку перед выдвижением, верно T2B**

T1O

Теперь возникло противоречие между T1 и обещаниями избирателей при бронировании — например:

- Бронируется заявка $(a, 2)$
- При бронировании избиратель p обещает не одобрять заявки с меньшими номерами
- Бронируется и выдвигается заявка $(b, 1)$
- По T1 избиратель p вынужден одобрить $(b, 1)$ вопреки обещанию

Чтобы исключить такие противоречия, «ослабим» T1:

T1O. Избиратель одобряет первую полученную заявку с номером, не меньшим всех забронированных (если ни один номер не забронирован, то просто первую полученную, как в T1), и отклоняет все заявки с меньшими номерами

Оптимизированное бронирование

Перед окончательной формулировкой действий отправителей и избирателей добавим следующую оптимизацию бронирования:

- Если избиратель ответил на бронирование номера $\mathfrak{n}$, то он не отвечает на бронирование номеров, меньших $\mathfrak{n}$
- Если избиратель получил заявку с номером $\mathfrak{n}$, то он не отвечает на бронирование номера $\mathfrak{n}$

С учётом такой оптимизации, избирателю достаточно хранить только один (самый большой) забронированный номер, игнорируя запросы на бронирование всех меньших

Утверждение. **Избиратель с оптимизацией бронирования и без неё одинаково одобряет и отклоняет заявки согласно T1O**

Восстановление данных после отказа

Для соблюдения T1O и T2B с учётом отказов и потери сообщений избиратель при выходе из строя с последующим восстановлением должен сохранять (восстанавливать)

- полученную им заявку с наибольшим номером и
- наибольший забронированный номер

Заявитель при выходе из строя с последующим восстановлением должен сохранять только номер последней заявки

Алгоритм Паксос: код заявителя

Переменные заявителя p:

- $n_p : \mathbb{N}_0 = 0$

Процедура выдвижения значения v заявителем p:

1. $n_p := n_p + 1$;
2. Отправить (**reserve**, $[p, n_p]$) всем избирателям
   - Это запрос на бронирование номера (p, n_p)
3. Принять (**ack**, x) от более чем половины избирателей
   - Это подтверждение бронирования, и x — это либо $\perp$, либо заявка
4. Если приняты только сообщения (**ack**, $\perp$), то отправить (**accept**, v, $[p, n_p]$) всем избирателям;
   иначе отправить им (**accept**, w, $[p, n_p]$) для значения w полученной заявки $x = (w, \mathfrak{m})$ с наибольшим номером $\mathfrak{m}$

Кроме того, в любой момент заявитель может принудительно завершить процедуру выдвижения значения («забросить» заявку) и выдвинуть новое значение (выполнить процедуру для нового v)

Алгоритм Паксос: код избирателя

Переменные избирателя p:

1. z_p — заявка или $\perp$, начальное значение $\perp$
2. $\mathfrak{n}_p$ — пара (идентификатор узла, число из $\mathbb{N}_0$) или $\perp$, начальное значение $\perp$

Действия после приёма ($\underline{\textbf{reserve}}$, $\mathfrak{n}$) избирателем p:

1. Если $\mathfrak{n}_p = \perp$ или $\mathfrak{n}_p < \mathfrak{n}$:
   1.1 Отправить (**ack**, z_p) в ответ
   1.2 $\mathfrak{n}_p := \mathfrak{n}$;

Действия после приёма ($\underline{\textbf{accept}}$, v, $\mathfrak{n}$) избирателем p:

1. Если $\mathfrak{n}_p = \perp$ или $\mathfrak{n}_p \leq \mathfrak{n}$:
   1.1 $z_p := (v, \mathfrak{n})$;
   1.2 $\mathfrak{n}_p := \mathfrak{n}$;

Алгоритм Паксос: итог

Утверждение (Д.з. 2). Если в некоторой конфигурации вычисления алгоритма избирателями выбрано значение v, то и во всех последующих конфигурациях выбирается только значение v

Это утверждение означает, в числе прочего, что выбор значения v избирателями — это монотонное свойство конфигураций

Значит, для достижения консенсуса достаточно время от времени собирать снимок сети в каком-либо узле (или в нескольких узлах, или во всех узлах), проверять по снимку, выбрано ли какое-либо значение, и рассылать это значение во все узлы для принятия решения

Алгоритм Паксос с наблюдателями

Но для *несколько более* эффективного принятия решения в алгоритм Паксос добавлена роль наблюдателя:

- Всякий раз, когда избирателем p впервые одобрено значение v, пара (v, p) отправляется заранее заданному (параметром алгоритма) числу произвольно выбранных наблюдателей
  - Например, всем наблюдателям
- Наблюдатель, получив пару (v, p), рассылает её всем остальным наблюдателям
- Если из полученных сообщений следует, что большинство избирателей получило значение v, то это значение рассылается всем узлам для принятия решения

Алгоритм Паксос с наблюдателями

Д.з. 3. Положим, что сообщения не теряются, выходить из строя может не более t узлов и заявитель изначально выдвигает значение своей входной переменной. Какое наименьшее число (а) заявителей, (б) избирателей, (в) наблюдателей и (г) узлов в целом требуется, чтобы алгоритм Паксос обладал свойствами единогласия и невырожденности? Ответ обосновать.

Д.з. 4. Положим, что сообщения не теряются, узлы не выходят из строя и в сети есть два заявителя и пять избирателей. Предложить способ нумерации заявок, и для него — бесконечное справедливое вычисление алгоритма Паксос с наблюдателями или без них, в котором решение не принимается (консенсус не достигается).