Алгебра от STEEM & GOLOS ч.1: аккаунт и сила голоса

Вступление

Ох уже эти блокчейны и опенсорсы с приписками о том, что код есть документация... Да и ладно, ну будем показывать пальцами, а попытаемся разобраться и отыскать еще одну крупицу истины.

В данном цикле статей я хочу поделиться своими изысканиям на тему расчета вознаграждений в блокчейнах STEEM и GOLOS, поведать о различиях, описать некоторые спорные моменты и возможно дать свою оценку происходящему. Вся информация будет разбита на несколько постов для более легко усваивания, в конце всего цикла будет предложен класс с методами расчетов всей описанной математики. Данный класс будет частью репозитория OnePlace, в процессе работы над которым и происходили данные изыскания.

На момент написании статьи работа велась с нодами STEEM 19.2 и GOLOS 16.4, указаний версий является важным аспектом, так как при последующих форках возможны изменения в расчетах, но это известно только наблюдателю из будущего. Примеры кода и формул будут представлены на языке JavaScript, а вызовы API в виде команд с использование CURL.

Как подобает хорошему пособию для начала разберем основные термины и несколько простых формул.

Аккаунт и его сила

Блокчейны STEEM и GOLOS крутятся вокруг идеи вознаграждения за создания и курирования контента, но это идея совершенно не

предполагает равенства между пользователям, как раз наоборот, тут поощеряется действия в виде увелечения силы вашего аккаунта, что по итогу влияет на объем вознаграждений, которые конкретный аккаунт может отчеканить для какого либо контента внутри сети. Разберем эти силы подробнее.

Описание терминов

Переменные пораждаемые расчетами:

- **VESTS** - сила голоса, ваши акции в блокчейне, именно это и есть сила вашего аккаунта, величина определяет мощь вашего лайка и т.д. От этого значения на аккаунт вешаются ярлыки по типу "кит", "дельфин" и остальных обывателей аквамира. Конечно же GOLOS не смог бы спокойно спать без своего названия этого значения и с долей эгоцентризма он был переименован в **GESTS**. В публичных клиентах как правило не представлен в чистом виде и более того показатель является составным, так как возможно делегирование своей силы и получения силы от других аккаунтов, в частности на раннем этапе жизни аккаунта (делегирование есть только в STEEM).

- **STEEM POWER** - сила голоса, именно это отображается в кошельке публичных клиентов, данный показатель динамичен и имеет растущую тенденцию. Но не стоит обольщаться, этот рост всего лишь отражение влияния инфляции, а не увеличения реальной силы голоса, ведь **VESTS** остается неизменным без каких либо действий со строны аккаунта.

- **steem_per_vest** - динамический показатель, ни что иное как коэффициент для расчета силы голоса (**SP**, оно же **STEEM POWER**). Коэффициент отражает инфляцию в системе и влияет на расчет **SP** самым прям образом. Данный

коэффициент можно увидеть на ресурсах steemd.com и golosd.com (открывается только в полнолуние). Стоит отметить - вы не найдете этого значение среди данных БЧ, оно также является результатом расчета.

Переменные получаемые непосредственно из БЧ и используемые в расчетах:

- **vesting_shares** - количество VESTS(GESTS), принадлежащих аккаунту
- **received_vesting_shares** - (STEEM) количество VESTS, делегированные текущему аккаунту
- **delegated_vesting_shares** - (STEEM) количество VESTS, делегированные другому аккаунту
- **total_vesting_fund_steem** - общее количество токенов (STEEM / GOLOS) в сети
- **total_vesting_shares** - Общее количество акций в системе (VESTS / GESTS) в сети

Формулы и расчеты

Разберем как нам получить **VESTS**, **steem_per_vests** и как переводить **VESTS** в **STEEM POWER** и наоборот.

Первым делом нам необходимо получить динамические показатели системы, а именно **total_vesting_fund_steem** и **total_vesting_shares**. Для этого достаточно обратиться к методу API блокчейнов **get_dynamic_global_properties**, эта процедура идентична для обоих БЧ:

```
curl -X POST -d
'{"id":1,"method":"get_dynamic_global_properties","pa
rams":[]}' https://ws.golos.io/
```

В ответе вы получите JSON объект с нужными переменными(**global_props**), но обратите внимание - это строки, необходимо отрезать постфиксы и перевести строки в числа с плавающей точкой.

```
// Расчет steem_per_vests
function steem_per_mvests(global_props){
  const total_vesting_fund_steem =
parseFloat(global_props.total_vesting_fund_steem.spli
t(' ')[0])
  const total_vesting_shares =
parseFloat(global_props.total_vesting_shares.split('
')[0])

  return 1e6 * total_vesting_fund_steem /
total_vesting_shares
}
```

Следующим шагом будет добыча заветного VESTS. Для этого воспользуемся методом API **get_accounts**:

```
curl -X POST -d
'{"id":1,"method":"get_accounts","params":
[["rusovds"]]}' https://ws.golos.io/
```

В ответе вы получите JSON массив, содержащий объект с пользовательскими данными(**account**), среди которых можно обнаружить заветные **vesting_shares**, **received_vesting_shares**, **delegated_vesting_shares**, используем их для расчета показателя **VESTS** конкретного аккаунта.

```
function calc_total_vests(account) {
```

```
  let total_vests =
parseFloat(account.vesting_shares.split(' ')[0])

  if (account.received_vesting_shares)
    total_vests +=
parseFloat(account.received_vesting_shares.split(' ')
[0])

  if (account.delegated_vesting_shares)
    total_vests -=
parseFloat(account.delegated_vesting_shares.split('
')[0])
}
```

Обратите внимание на условные операторы, в GOLOS нет переменных **received_vesting_shares**, **delegated_vesting_shares**. Видимо этот функционал появился после форка.

```
// Конвертация VESTS в STEEM POWER
function vests_to_sp(vests, steem_per_vest) {
  return vests / 1e6 * steem_per_mvests
}

// Конвертация STEEM POWER в VESTS
function sp_to_vests(sp, steem_per_vest) {
  return sp * 1e6 / steem_per_mvests
}
```

Голос, его вес, энергия голоса

Описание терминов

- **vote** - объект голоса(лайка), представляет из себя набор свойств, описывающие сколько и когда было отданно

"акций", каким конкретно аккаунтом, конкретному посту или комментарию;
- **active_votes** - массив объектов **vote**, описывающий все голоса за пост или комментарий;
- **rshares** - количество акций голоса, отданного аккаунтом за пост или комментарий, является свойством объекта **vote**. Это свойство по существу и есть сумма вознаграждения за конкретный лайк, но пока что не в человекочитаемом виде;
- **weight** - вес голоса, это процент который устанавливает сам пользователь в момент голосования;
- **voting_power** - процент доступной силы голоса. Данное свойство описывает какую часть от **VESTS** аккаунт может выделить при голосовании за пост. Значение стремится к нулю при частом голосовании и восстанавливается с истечением некоторого времени;
- **vote_power_reserve_rate** (STEEM) - количество голосов, которые аккаунт может регенирировать в течение суток;
- **vote_regeneration_per_day** (GOLOS) - тоже самое, что и **vote_power_reserve_rate**;
- **STEEM_100_PERCENT** - константа, равная 10000, что представляет из себя 100% от чего либо внутри блокчейна;
- **STEEMIT_VOTE_REGENERATION_SECONDS** - константа, количество секунд необходимое для восстановления **voting_power** с 0 до 10000 (100%);
- **max_vote_denom** - количество голосов, доступное пользователю в рамках **voting_power** и алгоритма его восстановления;
- **used_power** - используемая сила в голосовании, учитывающая **weight** голоса и **voting_power** и **max_vote_denom**.

Формулы и расчеты

Для начала получим необходимые переменные из БЧ:

- **voting_power** находиться в **account**, метод API **get_accounts** (см. выше);
- **STEEMIT_VOTE_REGENERATION_SECONDS** и **STEEM_100_PERCENT**, как и остальные константы, можно получить через метода API **get_config**

```
curl -X POST -d
'{"id":1,"method":"get_config","params":[]}'
https://ws.golos.io/
```

- **vote_regeneration_per_day** или **vote_power_reserve_rate** можно получить через метод API **get_dynamic_global_properties** (см. выше).

Теперь можно расчитать rshares будущего голоса аккаунта:

```
/*
  Расчет max_vote_denom

  Учтите, что vote_power_reserve_rate для GOLOS
  должен быть заменен на vote_regeneration_per_day
*/
function calc_max_vote_denom() {
  const seconds_in_day = 60 * 60 * 24
  return vote_power_reserve_rate *
STEEMIT_VOTE_REGENERATION_SECONDS / seconds_in_day
}

// Расчет used_power
function calc_used_power(account, weight) {
  const voting_power = account.voting_power
  const max_vote_denom = calc_max_vote_denom()
  return ((voting_power * weight / STEEM_100_PERCENT)
+ max_vote_denom - 1) / max_vote_denom
```

```
}

// Расчет rshares конретного голоса
function calc_rshares(account, weigth = 10000) {
  const total_vests = calc_total_vests(account) //
функция описаны в предыдущем разделе
  const max_vote_denom = calc_max_vote_denom()
  const used_power = calc_used_power(account, weight)
  const rshares = total_vests * used_power /
STEEM_100_PERCENT

  return Math.floor(rshares * 1e6)
}
```

Таким образом мы можем получить **rshares** голоса любого аккаунта, это только первая стадия расчетов и понимания реальной цены вашего голоса. **rshares** выражается большим целым числом и требует перевода в человекочитаемый вид.

Список вопросов, которые будут разобраны в следующем посте:

- rshares - сколько это в фиате?
- Что такое GBG и SBD?
- Как влияет весь массив active_votes на общую сумму выплат? Что такое квадратичность? (GOLOS)
- Как влияет первая и вторая выплаты на расчеты фиата? (GOLOS)
- Что делать если выплаты уже прошли, а динамические показатели системы, нужные для расчетов, сменились? Как заглянуть в прошлое? (GOLOS)

Благодарю за внимание!