Module sui_system::voting_power

Deprecated. Use VotingPowerInfoV2 instead.

Set total_voting_power as 10_000 by convention. Individual voting powers can be interpreted as easily understandable basis points (e.g., voting_power: 100 = 1%, voting_power: 1 = 0.01%) rather than opaque quantities whose meaning changes from epoch to epoch as the total amount staked shifts. Fixing the total voting power allows clients to hardcode the quorum threshold and total_voting power rather than recomputing these.

Quorum threshold for our fixed voting power--any message signed by this much voting power can be trusted up to BFT assumptions

Set the voting power of all validators. Each validator's voting power is initialized using their stake. We then attempt to cap their voting power at MAX_VOTING_POWER . If MAX_VOTING_POWER is not a feasible cap, we pick the lowest possible cap.

Create the initial voting power of each validator, set using their stake, but capped using threshold. We also perform insertion sort while creating the voting power list, by maintaining the list in descending order using voting power. Anything beyond the threshold is added to the remaining_power, which is also returned.

Insert new_info to info_list as part of insertion sort, such that info_list is always sorted using stake, in descending order.

Distribute remaining_power to validators that are not capped at threshold.

Update validators with the decided voting power.

Check a few invariants that must hold after setting the voting power.

Return the (constant) total voting power

Return the (constant) quorum threshold

Struct

Deprecated. Use VotingPowerInfoV2 instead.

```bash
```

```bash
```

Set total_voting_power as 10_000 by convention. Individual voting powers can be interpreted as easily understandable basis points (e.g., voting_power: 100 = 1%, voting_power: 1 = 0.01%) rather than opaque quantities whose meaning changes from epoch to epoch as the total amount staked shifts. Fixing the total voting power allows clients to hardcode the quorum threshold and total_voting power rather than recomputing these.

```bash
```

Quorum threshold for our fixed voting power--any message signed by this much voting power can be trusted up to BFT assumptions

```bash
```

```bash
```

```bash
```

```bash

```

```bash

```

```bash

```

Set the voting power of all validators. Each validator's voting power is initialized using their stake. We then attempt to cap their voting power at MAX_VOTING_POWER . If MAX_VOTING_POWER is not a feasible cap, we pick the lowest possible cap.

```
bash (package)
```

```
bash (package)
```

Create the initial voting power of each validator, set using their stake, but capped using threshold. We also perform insertion sort while creating the voting power list, by maintaining the list in descending order using voting power. Anything beyond the threshold is added to the remaining_power, which is also returned.

```bash

```

```bash

```

```
bash (package)
```

```
bash (package)
```

Insert new_info to info_list as part of insertion sort, such that info_list is always sorted using stake, in descending order.

```bash

```

```bash

```

Distribute remaining_power to validators that are not capped at threshold.

```bash

```

```bash

```

Update validators with the decided voting power.

```bash

```

```bash

```

Check a few invariants that must hold after setting the voting power.

```bash
```

```

```bash
```

Return the (constant) total voting power

```bash
```

```bash
```

Return the (constant) quorum threshold

```bash
```

```bash
```

Struct

```bash
```

Set total_voting_power as 10_000 by convention. Individual voting powers can be interpreted as easily understandable basis points (e.g., voting_power: 100 = 1%, voting_power: 1 = 0.01%) rather than opaque quantities whose meaning changes from epoch to epoch as the total amount staked shifts. Fixing the total voting power allows clients to hardcode the quorum threshold and total_voting power rather than recomputing these.

```bash
```

Quorum threshold for our fixed voting power--any message signed by this much voting power can be trusted up to BFT assumptions

```bash
```

```bash
```

```bash
```

```bash
```

```bash
```

```bash
```
```

Set the voting power of all validators. Each validator's voting power is initialized using their stake. We then attempt to cap their voting power at MAX_VOTING_POWER . If MAX_VOTING_POWER is not a feasible cap, we pick the lowest possible cap.

```
bash (package)
```

```
bash (package)
```

Create the initial voting power of each validator, set using their stake, but capped using threshold. We also perform insertion sort while creating the voting power list, by maintaining the list in descending order using voting power. Anything beyond the threshold is added to the remaining_power, which is also returned.

```bash
```

```bash
```

```
bash (package)
```

```
bash (package)
```

Insert new_info to info_list as part of insertion sort, such that info_list is always sorted using stake, in descending order.

```bash
```

```bash
```

Distribute remaining_power to validators that are not capped at threshold.

```bash
```

```bash
```

Update validators with the decided voting power.

```bash
```

```bash
```

Check a few invariants that must hold after setting the voting power.

```bash
```

```bash
```

Return the (constant) total voting power

```bash
```

```bash

```

Return the (constant) quorum threshold

```bash

```

```bash

```

Constants

Set total_voting_power as 10_000 by convention. Individual voting powers can be interpreted as easily understandable basis points (e.g., voting_power: 100 = 1%, voting_power: 1 = 0.01%) rather than opaque quantities whose meaning changes from epoch to epoch as the total amount staked shifts. Fixing the total voting power allows clients to hardcode the quorum threshold and total_voting power rather than recomputing these.

```bash

```

Quorum threshold for our fixed voting power--any message signed by this much voting power can be trusted up to BFT assumptions

```bash

```

```bash

```

```bash

```

```bash

```

```bash

```

```bash

```

Set the voting power of all validators. Each validator's voting power is initialized using their stake. We then attempt to cap their voting power at MAX_VOTING_POWER . If MAX_VOTING_POWER is not a feasible cap, we pick the lowest possible cap.

bash (package)

bash (package)

Create the initial voting power of each validator, set using their stake, but capped using threshold. We also perform insertion sort while creating the voting power list, by maintaining the list in descending order using voting power. Anything beyond the threshold is added to the remaining_power, which is also returned.

```bash

```

```bash
```

```
bash (package)
```

bash (package)

Insert new_info to info_list as part of insertion sort, such that info_list is always sorted using stake, in descending order.

```bash
```

```bash
```

Distribute remaining_power to validators that are not capped at threshold.

```bash
```

```bash
```

Update validators with the decided voting power.

```bash
```

```bash
```

Check a few invariants that must hold after setting the voting power.

```bash
```

```bash
```

Return the (constant) total voting power

```bash
```

```bash
```

Return the (constant) quorum threshold

```bash
```

```bash
```

Function

Set the voting power of all validators. Each validator's voting power is initialized using their stake. We then attempt to cap their voting power at MAX_VOTING_POWER . If MAX_VOTING_POWER is not a feasible cap, we pick the lowest possible cap.

```
bash (package)
```

```
bash (package)
```

Create the initial voting power of each validator, set using their stake, but capped using threshold. We also perform insertion sort while creating the voting power list, by maintaining the list in descending order using voting power. Anything beyond the threshold is added to the remaining_power, which is also returned.

```bash
```

```bash
```

```
bash (package)
```

```
bash (package)
```

Insert new_info to info_list as part of insertion sort, such that info_list is always sorted using stake, in descending order.

```bash
```

```bash
```

Distribute remaining_power to validators that are not capped at threshold.

```bash
```

```bash
```

Update validators with the decided voting power.

```bash
```

```bash
```

Check a few invariants that must hold after setting the voting power.

```bash
```

```bash
```

Return the (constant) total voting power

```bash

```

```bash

```

Return the (constant) quorum threshold

```bash

```

```bash

```

Function

Create the initial voting power of each validator, set using their stake, but capped using threshold. We also perform insertion sort while creating the voting power list, by maintaining the list in descending order using voting power. Anything beyond the threshold is added to the remaining_power, which is also returned.

```bash

```

```bash

```

```
bash (package)
```

```
bash (package)
```

Insert new_info to info_list as part of insertion sort, such that info_list is always sorted using stake, in descending order.

```bash

```

```bash

```

Distribute remaining_power to validators that are not capped at threshold.

```bash

```

```bash

```

Update validators with the decided voting power.

```bash

```

```bash

```

Check a few invariants that must hold after setting the voting power.

```bash
```

```bash
```

Return the (constant) total voting power

```bash
```

```bash
```

Return the (constant) quorum threshold

```bash
```

```bash
```

Function

`bash (package)`

`bash (package)`

Insert new_info to info_list as part of insertion sort, such that info_list is always sorted using stake, in descending order.

```bash
```

```bash
```

Distribute remaining_power to validators that are not capped at threshold.

```bash
```

```bash
```

Update validators with the decided voting power.

```bash
```

```bash
```

Check a few invariants that must hold after setting the voting power.

```bash
```

```

```bash
```

Return the (constant) total voting power

```bash
```

```bash
```

Return the (constant) quorum threshold

```bash
```

```bash
```

Function

Insert new_info to info_list as part of insertion sort, such that info_list is always sorted using stake, in descending order.

```bash
```

```bash
```

Distribute remaining_power to validators that are not capped at threshold.

```bash
```

```bash
```

Update validators with the decided voting power.

```bash
```

```bash
```

Check a few invariants that must hold after setting the voting power.

```bash
```

```bash
```
```

Return the (constant) total voting power

```bash
```

```bash
```

Return the (constant) quorum threshold

```bash
```

```bash
```

Function

Distribute remaining_power to validators that are not capped at threshold.

```bash
```

```bash
```

Update validators with the decided voting power.

```bash
```

```bash
```

Check a few invariants that must hold after setting the voting power.

```bash
```

```bash
```

Return the (constant) total voting power

```bash
```

```bash
```

Return the (constant) quorum threshold

```bash
```

```bash
```

Function

Update validators with the decided voting power.

```bash
```

```bash
```

Check a few invariants that must hold after setting the voting power.

```bash
```

```bash
```

Return the (constant) total voting power

```bash
```

```bash
```

Return the (constant) quorum threshold

```bash
```

```bash
```

Function

Check a few invariants that must hold after setting the voting power.

```bash
```

```bash
```

Return the (constant) total voting power

```bash
```

```bash
```

```

Return the (constant) quorum threshold

```bash

```

```bash

```

Function

Return the (constant) total voting power

```bash

```

```bash

```

Return the (constant) quorum threshold

```bash

```

```bash

```

Function

Return the (constant) quorum threshold

```bash

```

```bash

```
```