Incorrect Global `rewardsPerSecond` Accounting

```
 /**
     * @dev Add a new token to the reward pool system
     * @param _allocPoint Allocation points for the new pool
     * @param _depFee Deposit fee in basis points (1 = 0.01%)
     * @param _token Token to be staked
     * @param _withUpdate Whether to update all pools
     * @param _lastRewardTime Last reward time for this pool
     */
    function add(uint256 _allocPoint, uint256 _depFee, address _token, bool
_withUpdate, uint256 _lastRewardTime)
        public
        onlyOperator
    {
        checkPoolDuplicate(IERC20(_token));
        require(_depFee < 200, "GenesisRewardPool: deposit fee too high"); //
deposit fee can't be more than 2%

        if (_withUpdate) {
            massUpdatePools();
        }

        if (block.timestamp < poolStartTime) {
            // chef is sleeping
            if (_lastRewardTime == 0) {
                _lastRewardTime = poolStartTime;
            } else {
                if (_lastRewardTime < poolStartTime) {
                    _lastRewardTime = poolStartTime;
                }
            }
        } else {
            // chef is cooking
            if (_lastRewardTime == 0 || _lastRewardTime < block.timestamp) {
                _lastRewardTime = block.timestamp;
            }
        }
        bool _isStarted = (_lastRewardTime <= poolStartTime) || (_lastRewardTime <=
block.timestamp);

        uint256 pid = poolInfo.length;

        poolInfo.push(
            PoolInfo({
                token: IERC20(_token),
                depFee: _depFee,
```

```
                allocPoint: _allocPoint,
                poolRewardPerSec: _allocPoint,
                lastRewardTime: _lastRewardTime,
                accRewardsPerShare: 0,
                isStarted: _isStarted,
                totalStaked: 0
            })
        );

        if (_isStarted) {
            totalAllocPoint = totalAllocPoint.add(_allocPoint);
            rewardsPerSecond = rewardsPerSecond.add(_allocPoint);
        }

        emit PoolAdded(pid, _token, _allocPoint, _depFee);
    }
```

Code:

```
if (_isStarted) {
    totalAllocPoint = totalAllocPoint.add(_allocPoint);
    rewardsPerSecond = rewardsPerSecond.add(_allocPoint);  // ❌ wrong
}
```

Problem:

- In most **MasterChef-style staking contracts**, `rewardsPerSecond` (or `rewardsPerBlock`) is **global and fixed** (decided by protocol).
- Pool reward rates should be calculated **relative to allocPoints**:

  ```
  poolReward = rewardsPerSecond * pool.allocPoint / totalAllocPoint
  ```

- But here, each time a pool is added, the code **directly adds `_allocPoint` into `rewardsPerSecond`**.
- This means global emission rate keeps **growing with every pool**, instead of staying fixed and being divided among pools.

Example Walkthrough

Suppose:

- Initial `rewardsPerSecond = 10 tokens/sec`.

- Add **Pool 1** with `_allocPoint = 100`.

Expected:

- `rewardsPerSecond` stays **10**.
- Pool 1 gets **100 / 100 = 100%** → 10 tokens/sec.

Actual:

- `rewardsPerSecond = 0 + 100 = 100` (inflated).
- Pool 1 gets 100 tokens/sec (10x the intended reward).

Now add **Pool 2** with `_allocPoint = 100`.

Expected:

- Still 10 tokens/sec globally.
- Each pool gets 5 tokens/sec.

Actual:

- `rewardsPerSecond = 100 + 100 = 200`.
- Pool 1 gets ~100/sec, Pool 2 gets ~100/sec.
- **200 tokens/sec emitted instead of 10**.

Every pool increases emissions instead of dividing them.

Impact

- **Severe inflation of reward token supply.**
- Protocol could **run out of rewards early**.
- Pools receive **much higher rewards than intended**, breaking tokenomics.
- Attacker or opportunistic stakers can farm **much more than fair share**.

This is **critical** if the staking contract is live with finite reward reserves.

Recommended Fix

Do **not modify `rewardsPerSecond` when adding pools**.
Instead, use `totalAllocPoint` to proportionally distribute:

```
if (_isStarted) {
    totalAllocPoint = totalAllocPoint.add(_allocPoint);
}
```

Then when updating pool rewards:

```
uint256 poolReward = rewardsPerSecond * pool.allocPoint / totalAllocPoint;
```
