memory_buster

March 28, 2021

1 cadCAD memory buster

Danilo Lessa Bernardineli

An cadCAD model for frying the thread memory.

For every timestep, an constant number of random points are appended to an state variable array. This can be useful for diagonistics

```
[1]: from cadCAD.configuration.utils import config_sim
    from cadCAD.configuration import Experiment
    from cadCAD.engine import ExecutionMode, ExecutionContext
    from cadCAD.engine import Executor
    from cadCAD import configs
    import pandas as pd
    import numpy as np
    import plotly.express as px
    import sys
    import psutil
    import os
```

1.1 Model

```
[2]: initial_state = {
    # List that is going to bust the memory
    'things': np.array([0.0, 1.0]),

# RAM used by the state variables
    'state_bytes': None,

# RAM used by the state history
    'history_bytes': None,

# RAM used by the process
    'process_bytes': None,

# Hack variable that is always nothing
```

```
'nothing': None
}

system_params = {
    # Number of random numbers to append to a list each timestep
    "amount_of_random_values": [int(1e4)],
    "erase_history": [True, False]
}
```

1.1.1 Logic

```
[3]: def s_add_things(params, substep, history, previous_state, policy_input):
         Add random numbers to the 'things' state variable
         things = previous_state['things']
         N = params['amount_of_random_values']
         values = np.random.rand(N)
         things = np.append(things, values)
         return ('things', things)
     def s_measure_history(params, substep, history, previous_state, policy_input):
         Measure how much memory the 'things' state variable history is using
         historic_bytes = [sys.getsizeof(substep_state.get('things', 0))
                           for timestep_state in history
                           for substep_state in timestep_state]
         used_bytes = sum(historic_bytes)
         return ('history_bytes', used_bytes)
     def s_measure_state(params, substep, history, previous_state, policy_input):
         Measure how much memory the 'things' state variable is using
         used_bytes = sys.getsizeof(previous_state['things'])
         return ('state_bytes', used_bytes)
     def s_measure_process(params, substep, history, previous_state, policy_input):
         Get the current process and see how much memory is on it
         process = psutil.Process(os.getpid())
         used_bytes = process.memory_info().data
```

```
return ('process_bytes', used_bytes)

def s_erase_history(params, substep, history, previous_state, policy_input):
    """
    Iterate on the history and erase everything
    """
    if params['erase_history'] is True:
        for timestep_state in history:
            for substep_state in timestep_state:
                 substep_state['things'] = None
    return ('nothing', None)
```

1.1.2 Structure

```
[4]: partial_state_update_blocks = [
       {
           # Add lots of random numbers to the simulation
           'policies': {},
           'variables': {
               'things': s_add_things
       },
           # Measure memory usage
           'policies': {},
           'variables': {
               'state_bytes': s_measure_state,
               'history_bytes': s_measure_history,
               'process_bytes':s_measure_process
           }
       },
           # Erase history of the variables
           'policies': {},
           'variables': {
               'nothing': s_erase_history
       },
     ]
```

Actual cadCAD usage

```
[5]: sim_config = config_sim({
    "N": 1, # the number of times we'll run the simulation ("Monte Carlo runs")
    "T": range(200), # the number of timesteps the simulation will run for
    "M": system_params # the parameters of the system
```

```
})
del configs[:] # Clear any prior configs
experiment = Experiment()
experiment.append_configs(
    initial_state = initial_state,
    partial_state_update_blocks = partial_state_update_blocks,
    sim_configs = sim_config
)
exec_context = ExecutionContext()
simulation = Executor(exec_context=exec_context, configs=configs)
raw_result, tensor_field, sessions = simulation.execute()
/ /__/ /_/ / /_/ / /__/ ___ \/ /_/ /
\___/\__,_/\__,_/\___/_/ |_/___/
by cadCAD
Execution Mode: local_proc
Configuration Count: 1
Dimensions of the first simulation: (Timesteps, Params, Runs, Vars) = (200, 2,
Execution Method: local_simulations
SimIDs
       : [0, 0]
SubsetIDs: [0, 1]
```

: [0, 1] Ns ExpIDs : [0, 0]

Execution Mode: parallelized Total execution time: 11.14s

```
[6]: df = pd.DataFrame(raw_result)
     # Put parameters inside the data frame
     df = df.assign(**configs[0].sim_config['M'])
     for i, (_, n_df) in enumerate(df.groupby(['simulation', 'subset', 'run'])):
         df.loc[n_df.index] = n_df.assign(**configs[i].sim_config['M'])
     df.head(10)
```

```
[6]: things state_bytes history_bytes process_bytes nothing simulation \
                                                                  None
         None
                         {\tt NaN}
                                          {\tt NaN}
                                                          {\tt NaN}
                                                                  None
     1
         None
                         NaN
                                          NaN
                                                          NaN
                                                                                   0
```

```
2
               80120.0
                                 120.0
                                           211955712.0
                                                           None
                                                                            0
    None
3
    None
               80120.0
                                 120.0
                                           211955712.0
                                                           None
                                                                            0
4
    None
                                                           None
                                                                            0
               80120.0
                                 120.0
                                           211955712.0
                                                                            0
5
    None
              160120.0
                              240376.0
                                           212353024.0
                                                           None
6
    None
              160120.0
                              240376.0
                                           212353024.0
                                                           None
                                                                            0
7
    None
              160120.0
                              240376.0
                                           212353024.0
                                                           None
                                                                            0
    None
              240120.0
                              480424.0
                                                           None
                                                                            0
8
                                           213073920.0
9
    None
              240120.0
                              480424.0
                                           213073920.0
                                                           None
                                                                            0
   subset
                 substep
                          timestep
                                      amount_of_random_values
                                                                 erase_history
0
              1
                       0
                                  0
                                                         10000
                                                                           True
        0
              1
                       1
                                  1
                                                         10000
1
        0
                                                                           True
                       2
2
        0
              1
                                  1
                                                                           True
                                                         10000
3
        0
              1
                       3
                                   1
                                                         10000
                                                                           True
4
        0
              1
                        1
                                   2
                                                         10000
                                                                           True
                        2
                                  2
5
        0
              1
                                                         10000
                                                                           True
6
        0
              1
                        3
                                   2
                                                         10000
                                                                           True
7
        0
                                   3
              1
                        1
                                                         10000
                                                                           True
        0
                        2
                                   3
8
              1
                                                         10000
                                                                           True
9
        0
              1
                       3
                                   3
                                                         10000
                                                                           True
```